DOCKET NO: 230421US26YA

IN THE UNITED STATES PATENT & TRADEMARK OFFICE

| | |
|---|---|
| IN RE APPLICATION OF | : |
| ANDREJ S. MITROVIC | : EXAMINER: SAXENA, AKASH |
| SERIAL NO: 10/673,501 | : |
| FILED: SEPTEMBER 30, 2003 | : GROUP ART UNIT: 2128 |
| FOR: SYSTEM AND METHOD FOR USING FIRST-PRINCIPLES SIMULATION TO CHARACTERIZE A SEMICONDUCTOR MANUFACTURING PROCESS | : |

**APPEAL BRIEF UNDER 37 CFR 41.37**

COMMISSIONER FOR PATENTS
ALEXANDRIA, VIRGINIA 22313

SIR:

This is an appeal of the final Office Action dated February 27, 2008. A Notice of Appeal was filed on June 17, 2008.

## TABLE OF CONTENTS

## I. 41.37(C)(1)(I) Real Party of Interest

The real party of interest in this appeal is the assignee Tokyo Electron Limited whose address is Akasaka Biz Tower, 3-1, Akasaka 5-chome, Minato-ku, Tokyo 107-6325, Japan.

## II. 41.37(C)(1)(II) Related Appeals and Interferences

There are no related interferences. There are related appeals filed or to be filed in U.S. Serial Nos. 10/673,138; 10/673,467; 10/673,506; 10/673,507; and 10/673,583.

## III. 41.37(C)(1)(III) Status of Claims

Claims 1-44 and 48-50 are pending and appealed. Claims 45-47 and 51 are canceled.

Claim 1 stands provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over Claim 1 copending Application No. 10/673,583; Claim 1 stands provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over Claim 1 of copending Application No. 10/673,138; Claim 1 stands provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over Claim 1 of copending application No. 10/673,507; Claims 1-51 stand rejected under 35 U.S.C. § 112, first paragraph, as based on a disclosure which is not enabling, Claims 1-51 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,802,045 issued to Sonderman et al, in view of IEEE article "Mathematic-physical engine: parallel processing for modeling and simulation of physical phenomena" by Jain et al.

## IV. 41.37(c)(1)(iv) Status of Amendments

An amendment was filed for this application on November 2, 2007 which resulted in the final Office Action dated February 27, 2008. An amendment after the final rejection was

filed on May 27, 2008 canceling Claims 45-47 and 51. A terminal disclaimer was also filed on May 27, 2008. The terminal disclaimer was approved June 2, 2008.

## V. 41.37(c)(1)(v) Summary of Claimed Subject Matter

**Claim 1**, the first of the independent claims appealed, will be treated as a picture claim representing many of the features in the remaining independent claims. Accordingly, a claim chart for support is provided below showing support from the specification for the claim elements.

In short, Claim 1 defines a method of controlling a process performed by a semiconductor processing tool. The method inputs process data ***relating to an actual process being performed*** by the semiconductor processing tool, and inputs a first principles physical model including a set of computer-encoded differential equations. The first principles physical model describes at least one of a basic physical or chemical attribute of the semiconductor processing tool. The method performs first principles simulation ***for the actual process being performed during performance of the actual process*** using the physical model to provide a first principles simulation result in accordance with the process data relating to the actual process being performed in order to simulate the actual process being performed. The first principles simulation result is ***produced in a time frame shorter in time than the actual process being performed***. The model uses the simulation result as part of a data set that characterizes the actual process being performed by the semiconductor processing tool.

Accordingly, Claim 1 makes clear that a first principles simulation result ***for the actual process being performed during performance of the actual process*** is used as part of a data set that characterizes the actual process being performed by the semiconductor processing tool. The following is a claim chart comparison of the claim elements to the

disclosure in the specification. Emphasis has been added for convenience in some of the longer passages from the specification.

| **Claim 1** | **Support in U.S. Pat. Appl. No. 10/673,501** |
|---|---|
| A method of controlling a process performed by a semiconductor processing tool, comprising | Specification, numbered paragraph [0011]: One aspect of the present invention is a method of facilitating a process performed by a semiconductor processing tool, which includes inputting data relating to a process performed by the semiconductor processing tool, and inputting a first principles physical model relating to the semiconductor processing tool. First principles simulation is then performed using the input data and the physical model to provide a simulation result for the process performed by the semiconductor processing tool, and the simulation result is used as part of a data set that characterizes the process performed by the semiconductor processing tool. |

| | |
|---|---|
| inputting process data relating to an actual process being performed by the semiconductor processing tool | Specification, numbered paragraph [0032]: Data input device 104 is a device for collecting data relating to a process performed by the semiconductor processing tool 102 ***and inputting the collected data to the first principles simulation processor 106.*** . . . In one embodiment, the data input device 104 may be implemented as a physical sensor for collecting data about the semiconductor processing tool 102 itself, and/or the environment contained within a chamber of the tool. Such data may include fluid mechanic data such as gas velocities and pressures at various locations within the process chamber, electrical data such as voltage, current, and impedance at various locations within the electrical system of the process chamber, chemical data such as specie concentrations and reaction chemistries at various locations within the process chamber, thermal data such as gas temperature, surface temperature, and surface heat flux at various locations within the process chamber, plasma processing data (when plasma is utilized) such as a plasma density (obtained, for example, from a Langmuir probe), an ion energy (obtained, for example, from an ion energy spectrum analyzer), and mechanical data such as pressure, deflection, stress, and strain at various locations within the process chamber.<br><br>Specification, numbered paragraph [0039]: FIG. 2 is a flow chart showing a process for using first principles simulation techniques to-facilitate a process performed by a semiconductor processing tool in accordance with an embodiment of the present invention. The process shown in FIG. 2 may be run on the first principles simulation processor 104 of FIG. 1, for example. As seen in FIG. 2, the process begins in step 201 with the inputting of data related to a process performed by the semiconductor processing tool 102. As discussed above, the input data may be data relating to physical attributes of the tool/tool environment and/or data relating to a process performed by the tool on a semiconductor wafer or results of such process. As also described above, ***the input data may be directly input from a physical sensor or metrology tool coupled to the first principles simulation processor 104***, or indirectly input from a manual input device or database. |

| | |
|---|---|
| inputting a first principles physical model including a set of computer-encoded differential equations, the first principles physical model describing at least one of a basic physical or chemical attribute of the semiconductor processing tool; | Specification, numbered paragraph [0035]: First principles physical model 106 is a model of the physical attributes of the tool and tool environment as well ***as the fundamental equations necessary to perform first principles simulation*** and provide a simulation result for facilitating a process performed by the semiconductor processing tool. Thus, the first principles physical model 106 depends to some extent on the type of semiconductor processing tool 102 analyzed as well as the process performed in the tool. For example, the physical model 106 may include a spatially resolved model of the physical geometry of the tool 102, which is different, for example, for a chemical vapor deposition (CVD) chamber and a diffusion furnace. Similarly, the first principles equations necessary to compute flow fields are quite different than those necessary to compute temperature fields. The physical model 106 may be a model as implemented in commercially available software, such as ANSYS, of ANSYS Inc., Southpointe, 275 Technology Drive Canonsburg, Pa. 15317, FLUENT, of Fluent Inc., 10 Cavendish Conn. Centerra Park, Lebanon, N.H. 03766, or CFD-ACE+, of CFD Research Corp., 215 Wynn Dr., Huntsville, Ala. 35805, to compute flow fields, electromagnetic fields, temperature fields, chemistry, surface chemistry (i.e. etch surface chemistry or deposition surface chemistry). However, special purpose or custom models developed from first principles to resolve these and other details within the processing system may also be used.<br><br>Specification, numbered paragraph [0036]: First principles simulations in the present invention include, but are not limited to, simulations of electro-magnetic fields derived from ***Maxwell's equations, continuum simulations, for example, for mass, momentum, and energy transport derived from continuity, the Navier-Stokes equation*** and the First Law of Thermodynamics, as well as atomistic simulations derived from the Boltzmann equation, such as for example Monte Carlo simulations of rarefied gases (see Bird, G. A. 1994. Molecular gas dynamics and the direct simulation of gas flows, Clarendon Press). |

| | |
|---|---|
| performing first principles simulation for the actual process being performed during performance of the actual process using the physical model | Specification, numbered paragraph [0012]: A first principles simulation processor is configured to input a first principles physical model relating to the semiconductor processing tool, and perform first principles simulation using the input data and the physical model to provide a first principles simulation result. The first principles simulation result is used as part of a data set that characterizes the process performed by the semiconductor processing tool.<br><br>Specification, numbered paragraph [0036]: First principles simulation processor 108 is a processing device that applies data input from the data input device 104 to the first principles physical model 108 to execute a first principles simulation. Specifically, the first principles simulation processor 108 may use the data provided by the data input device 104 to set initial conditions and/or boundary conditions for the first principles physical model 106, which is then executed by the simulation module.<br><br>Specification, numbered paragraph [0057]: In this embodiment, steady-state simulations are repeatedly run concurrently with the process by using the physical sensor measurements to repeatedly update boundary conditions of the first principles simulation model. |
| to provide a first principles simulation result in accordance with the process data relating to the actual process being performed in order to simulate the actual process being performed, | Specification, numbered paragraph [0036]: The output of the first principles simulation processor 108 is a simulation result that is used to facilitate a process performed by the semiconductor processing tool 102. For example, the simulation result may be used to facilitate process development, process control and fault detection as well as to provide virtual sensor outputs that facilitate tool processes, as will be further described below. |

| | |
|---|---|
| said first principles simulation result being produced in a time frame shorter in time than the actual process being performed; and | Specification, numbered paragraph [0041]: In step 205, the first principles simulation processor 108 uses the input data of step 201 and the first principles physical model of step 203 to execute a first principles simulation and provide a simulation result. Step 205 may be performed either concurrently with or not concurrently with the process performed by the semiconductor processing tool. For example, simulations that can be performed at short solution times may be run concurrently with a tool process, and results used to control the process. More computationally intensive simulations may be performed not concurrently with the tool process and the simulation result may be stored in a library for later retrieval. In one embodiment, step 205 includes using the input data of step 201 to set initial and/or boundary conditions for the physical model provided in step 205.<br><br>Specification, numbered paragraph [0057]: In this embodiment, steady-state simulations are repeatedly run concurrently with the process by using the physical sensor measurements to repeatedly update boundary conditions of the first principles simulation model. |

| | |
|---|---|
| using the simulation result as part of a data set that characterizes the actual process being performed by the semiconductor processing tool. | Specification, numbered paragraph [0012]: The simulation result is used as part of a data set that characterizes the process performed by the semiconductor processing tool.<br><br>Specification, numbered paragraph [0042]: Once the simulation is executed, the simulation result is used to facilitate a process performed by the semiconductor processing tool 102. As used herein, the term "facilitate a process performed by the semiconductor processing tool" includes using the simulation result for example to detect a fault in the process, to control the process, to characterize the process for manufacturing runs, to provide virtual sensor readings relating to the process, or any other use of the simulation result in conjunction with facilitating a process performed by the semiconductor processing tool 102.<br><br>Specification, numbered paragraph [0048]: The present inventors have also discovered that the network architecture of FIG. 3 provides the ability to distribute model results done at one processing tool 102 for one condition set, to other similar or identical tools operating later under the same or similar conditions, so redundant simulations are eliminated. Running simulations only for unique processing conditions at on-tool and standalone modules and re-using results from similar tools that have already known simulated solutions allows for rapid development of lookup libraries containing results that can be used for diagnostics and control over a large range of processing conditions. Further, the reuse of the known solutions as initial conditions for first principles simulation reduces the computational requirements and facilitates the production of simulated solutions in a time frame consistent with on-line control. Similarly, the network architecture of FIG. 3 also provides the ability to propagate changes and refinements made to physical models and model input parameters from one simulation module to others in the network. ***For example, if during process runs and parallel executions of a model it is determined that some input parameters need to be changed, then these changes can be propagated to all other simulation modules and tools via the network.*** |

**Claim 23** defines a system which is similar to the method of Claim 1. Thus, the

features of Claim 23 are supported in the specification by numbered paragraphs [0011], [0012], [0032]. [0035], [0036], [0039], [0041], [0042], [0048], and [0057].

**Claim 48** defines at least one of non-volatile media and volatile media containing program instructions for execution on a processor, which is similar to method Claim 1. Thus, the features of Claim 48 are supported in the specification by numbered paragraphs [0011], [0012], [0032]. [0035], [0036], [0039], [0041], [0042], [0048], and [0057].

## VI. 41.37(C)(1)(VI) Grounds of Rejection for Review

Whether the rejection of Claim 1 under the judicially created doctrine of obviousness-type double patenting over Claim 1 of U.S. Pat. Appl. No. 10/673,583 (the '583 application) should be reversed. Whether the rejection of Claim 1 under the judicially created doctrine of obviousness-type double patenting over Claim 1 of U.S. Pat. Appl. No. 10/673,138 (the '138 application) should be reversed. Whether the rejection of Claim under the judicially created doctrine of obviousness-type double patenting over Claim 1 of U.S. Pat. Appl. No. 10/673,507 (the '507 application) should be reversed. Whether the rejection of Claims 1-44 and 48-50 under 35 U.S.C. § 112, first paragraph, as being based on a non-enabling disclosure should be reversed. Whether the rejection of Claims 1-44 and 48-50 under 35 U.S.C. § 103(a) as being unpatentable over Sonderman et al in view of Jain et al should be reversed.

## VII. 41.37(C)(1)(VII) ARGUMENTS

### A. Regarding the 35 USC 112 1st Paragraph Rejection of Claims 1-44 and 48-50

Briefly recapitulating, Claim 1 defines a method of controlling a process performed by a semiconductor processing tool, including:

> 1) inputting process data relating to an actual process being performed by the semiconductor processing tool,
>
> 2) inputting a first principles physical model including a set of computer-encoded differential equations, the first principles physical model describing at least one of a basic physical or chemical attribute of the semiconductor processing tool,
>
> 3) performing first principles simulation for the actual process being performed ***during performance of the actual process*** using the physical model to provide a first principles simulation result in accordance with the process data relating to the actual process being performed in order to simulate the actual process being performed, ***said first principles simulation result being produced in a time frame shorter in time than the actual process being performed,*** and
>
> 4) using the simulation result as part of a data set that characterizes the actual process being performed by the semiconductor processing tool..

The claim defines clearly a process where data input from an actual process being performed is used for producing a first principles simulation result, produced for the actual process being performed during performance of the actual process. The result obtained is produced in a time frame shorter in time than the actual process being performed. The result obtained is then used as part of a data set that characterizes the actual process being performed by the semiconductor processing tool. M.P.E.P. 2164.01 states that the test of enablement is whether one reasonably skilled in the art could make or use the invention from the disclosures in the patent coupled with information known in the art ***without undue experimentation***. Appellant submits that details of 1) inputting the process data and 2) inputting the first principles physical model including what basic physical and chemical attribute of the semiconductor processing tool are used are disclosed in Appellant's filed specification. For instance, details of inputting data are given for example at numbered paragraphs [0032] and [0033], which state:

> Data input device 104 is a device for collecting data relating to a process performed by the semiconductor processing tool 102 and inputting the collected data to the first principles simulation processor 106. The process performed by the semiconductor process tool 102 may be a characterization process (i.e. process design or development), a cleaning process, a production process, or any other process performed by the semiconductor processing tool. In one embodiment, the data input device 104 may be implemented as a physical sensor for collecting data about the semiconductor processing tool 102 itself, and/or the environment contained within a chamber of the tool. Such data may include fluid mechanic data such as gas velocities and pressures at various locations within the process chamber, electrical data such as voltage, current, and impedance at various locations within the electrical system of the process chamber, chemical data such as specie concentrations and reaction chemistries at various locations within the process chamber, thermal data such as gas temperature, surface temperature, and surface heat flux at various locations within the process chamber, plasma processing data (when plasma is utilized) such as a plasma density (obtained, for example, from a Langmuir probe), an ion energy (obtained, for example, from an ion energy spectrum analyzer), and mechanical data such as pressure, deflection, stress, and strain at various locations within the process chamber.
>
> In addition to the tool and tool environment data, the data input device 104 may collect data relating to the process itself, or process results obtained on a semiconductor wafer that the tool 102 is performing a process on. In one embodiment, data input device 104 is implemented as a metrology tool coupled to the semiconductor processing tool 102. The metrology tool may be configured to measure process performance parameters such as: etch rate, deposition rate, etch selectivity (ratio of the rate at which a first material is etched to the rate at which a second material is etched), an etch critical dimension (e.g. length or width of feature), an etch feature anisotropy (e.g. etch feature sidewall profile), a film property (e.g. film stress, porosity, etc.), a mask (e.g. photoresist) film thickness, a mask (e.g. photoresist) pattern critical dimension, or any other parameter of a process performed by the semiconductor processing tool 102.

Details of inputting a first principles physical model for performing the first principles simulation result are given for example at numbered paragraphs [0035] and [0036] which state that:

> First principles physical model 106 is a model of the physical attributes of the tool and tool environment as well as the fundamental equations necessary to perform first principles simulation and provide a simulation result for facilitating a process performed by the semiconductor processing tool. Thus, the first principles physical model 106 depends to some extent on the type of semiconductor processing tool 102 analyzed as well as the process performed in the tool. For example, the physical model 106 may include a spatially resolved model of the physical geometry of the tool 102, which is

different, for example, for a chemical vapor deposition (CVD) chamber and a diffusion furnace. Similarly, the first principles equations necessary to compute flow fields are quite different than those necessary to compute temperature fields. The physical model 106 may be a model as implemented in commercially available software, such as ANSYS, of ANSYS Inc., Southpointe, 275 Technology Drive Canonsburg, PA 15317, FLUENT, of Fluent Inc., 10 Cavendish Ct. Centerra Park, Lebanon, NH 03766, or CFD-ACE+, of CFD Research Corp., 215 Wynn Dr., Huntsville, AL 35805, to compute flow fields, electro-magnetic fields, temperature fields, chemistry, surface chemistry (i.e. etch surface chemistry or deposition surface chemistry). However, special purpose or custom models developed from first principles to resolve these and other details within the processing system may also be used.

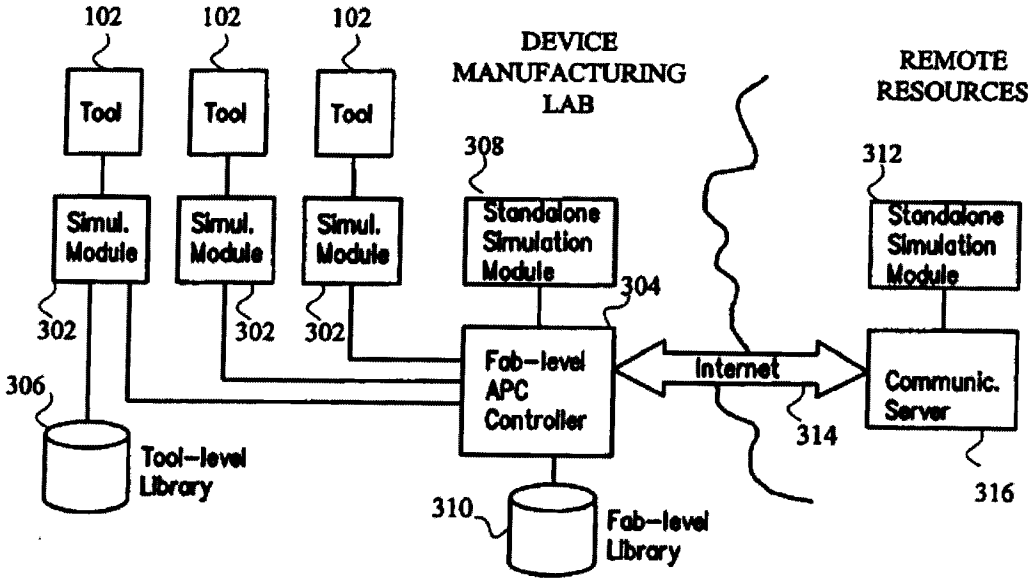
First principles simulation processor 108 is a processing device that applies data input from the data input device 104 to the first principles physical model 108 to execute a first principles simulation. Specifically, the first principles simulation processor 108 may use the data provided by the data input device 104 to set initial conditions and/or boundary conditions for the first principles physical model 106, which is then executed by the simulation module. First principles simulations in the present invention include, but are not limited to, simulations of electro-magnetic fields derived from Maxwell's equations, continuum simulations, for example, for mass, momentum, and energy transport derived from continuity, the Navier-Stokes equation and the First Law of Thermodynamics, as well as atomistic simulations derived from the Boltzmann equation, such as for example Monte Carlo simulations of rarefied gases (see Bird, G.A. 1994. Molecular gas dynamics and the direct simulation of gas flows, Clarendon Press). First principles simulation processor 108 may be implemented as a processor or workstation physically integrated with the semiconductor processing tool 102, or as a general purpose computer system such as the computer system 1401 of Figure 14. The output of the first principles simulation processor 108 is a simulation result that is used to facilitate a process performed by the semiconductor processing tool 102. For example, the simulation result may be used to facilitate process development, process control and fault detection as well as to provide virtual sensor outputs that facilitate tool processes, as will be further described below.

Thus, one of ordinary skill in the art, from the detailed information provided in the specification as to the data input and as to the commercially availability of software programs, would **not** have to use undue experimentation to apply the respective physical attributes that each model is tailored to accept in order to perform the claimed inputting a first principles physical model step.

The examiner requested details of the models which lead to the unexpected result of being able to avoid the lengthy time conventionally required for the generation of a first principles model simulation. See page 10 of the Office Action. Appellant points out the procedures by which the unexpected results of the invention are achieved. It is not the details of the models, but rather the details as to how the model calculations are implemented which reduce the time for calculations. For instance, the disclosed characteristics in the specification which permit simulation results to be obtained in a time frame compatible with using the first principles model simulation result for real time process control are enumerated below with reference to the numbered paragraphs in the filed specification:

> **1)** the use of interconnected resources inside a semiconductor device manufacturing facility to perform the first principles simulation (see specification, numbered paragraph [0043] and Figure 3, both reproduced below),
>
> **2)** the use of code parallelization among interconnected computational resources inside the semiconductor device manufacturing facility (see specification, numbered paragraphs [0047] and [0048] reproduced below),
>
> **3)** the sharing of simulation information among interconnected resources inside the semiconductor device manufacturing facility (see specification, numbered paragraphs [0047] and [0048] reproduced below), and
>
> **4)** the reduction in redundant execution of substantially similar first principles simulations by different resources the reuse of known solutions as initial conditions for the first principles simulation, as features which used singularly or in combination lead to a simulation result in a time frame consistent with real time process control in a semiconductor processing tool (see specification, numbered paragraphs [0047] and [0048] reproduced below).

**Figure 3**




**[0043]** FIG. 3 is a block diagram of a network architecture that may be used to provide first principles simulation techniques to facilitate a process performed by a semiconductor processing tool in accordance with an embodiment of the present invention. As seen in this figure, the network architecture includes a device manufacturing fab connected to remote resources via the Internet 314. The device manufacturing fab includes a plurality of semiconductor processing tools 102 connected to respective simulation modules 302. As described with respect to FIG. 1, each semiconductor processing tool 102 is a tool for performing a process related to manufacturing a semiconductor device such as an integrated circuit. Each simulation module 302 is a computer, workstation, or other processing device capable of executing first principles simulation techniques to facilitate a process performed by a semiconductor processing tool 102. Thus, each simulation module 302 includes the first principles physical model 106 and the first principles simulation processor 108 described with respect to FIG. 1, as well as any other hardware and/or software that may be helpful for executing first principles simulations. Moreover, simulation modules 302 are configured to communicate with the fab-level advanced process control (APC) controller using any known network communication protocol. Each simulation module 302 may be implemented as a general purpose computer such as the computer system 1401 of FIG. 14.

**[0047]** The present inventors have discovered that the network configuration of FIG. 3 provides computational and storage resource sharing that allows a broad range of first principles simulation results at reasonable solution speeds, thus providing meaningful on-tool simulation capabilities that

can facilitate processes performed by the tool. Specifically, while simple simulations may be executed by a tool's dedicated simulation module, complex simulations requiring greater computational resources may be executed using code parallelization techniques on multiple simulation modules in the network that may be on-tool or standalone. Even on-tool simulation modules in equipment currently under preventive maintenance may be used as a shared computational resource, provided there is power to the simulation module. Similarly, simulation results used for later lookup can be stored in libraries (e.g. storage devices) anywhere in the fab network, and accessed by all tools when lookups of diagnostic or control data are made.

**[0048]** The present inventors have also discovered that the network architecture of FIG. 3 provides the ability to distribute model results done at one processing tool 102 for one condition set, to other similar or identical tools operating later under the same or similar conditions, so redundant simulations are eliminated. Running simulations only for unique processing conditions at on-tool and standalone modules and re-using results from similar tools that have already known simulated solutions allows for rapid development of lookup libraries containing results that can be used for diagnostics and control over a large range of processing conditions. Further, the reuse of the known solutions as initial conditions for first principles simulation reduces the computational requirements and facilitates the production of simulated solutions in a time frame consistent with on-line control. Similarly, the network architecture of FIG. 3 also provides the ability to propagate changes and refinements made to physical models and model input parameters from one simulation module to others in the network. For example, if during process runs and parallel executions of a model it is determined that some input parameters need to be changed, then these changes can be propagated to all other simulation modules and tools via the network.

Hence, it is respectfully submitted that, in view of the disclosure of commercial software available for the different physical models disclosed in the specification and in view of the disclosure of procedures by which the time for producing a first principles model simulation result can be reduced, the invention does not require undue experimentation.

Accordingly, the 35 U.S.C. 112, first paragraph, rejection of Claims 1-44 and 48-50 should be reversed.

**B. Regarding the 35 USC 103 Rejection of Claims 1-44 and 48 over Sonderman et al and Jain et al**

The Office Action makes clear on pages 3 and 4 that the Examiner and the Appellant disagree as to whether the subscripts in Sonderman et al $S_i$ associated with the silicon wafer disclosure refers to process control for the same wafer being processed or process control for subsequent wafers. The Examiner's position is that Sonderman et al would have used $S_{i+1}$ to designate subsequent wafers.

Yet, Appellant respectfully points out that, at col. 9, lines 46-51, Sonderman et al specifically states:

> The system 100 ***then*** optimizes the simulation (described above) to find more optimal process target ($T_i$) for each silicon wafer, ***$S_i$ to be processed***. These target values are then used ***t***o generate ***new control inputs, $X_{Ti}$,*** on the line 805 to control ***a subsequent process of a silicon wafer $S_i$***. [Emphasis added]

The plain reading of this section of Sonderman et al is that the system 100 ***then*** (e.g., at time T1) optimizes the simulation for each silicon wafer, ***$S_i$ to be processed*** (e.g., later at time T2). In other words, the simulation results of Sonderman et al produce a new control input for each silicon wafer ***to be processed***. Thus, Appellant respectfully submits that Sonderman et al teach performing a simulation result for a process to be performed ***before*** performance of the actual process, and do **not** teach the claimed performing first principles simulation ***for the actual process being performed during performance of the actual process***.[1]

Other sections of Sonderman et al support Appellant's position on this matter that the simulation results in Sonderman et al are made prior to controlling a subsequent process. For instance, Figure 4 of Sonderman et al (reproduced below) shows that the simulation results are produced ***ahead of performing a process*** and thus have to be based on historical data.

[1] Appellant also point out that Sonderman et al do not disclose or suggest a first principles simulation.

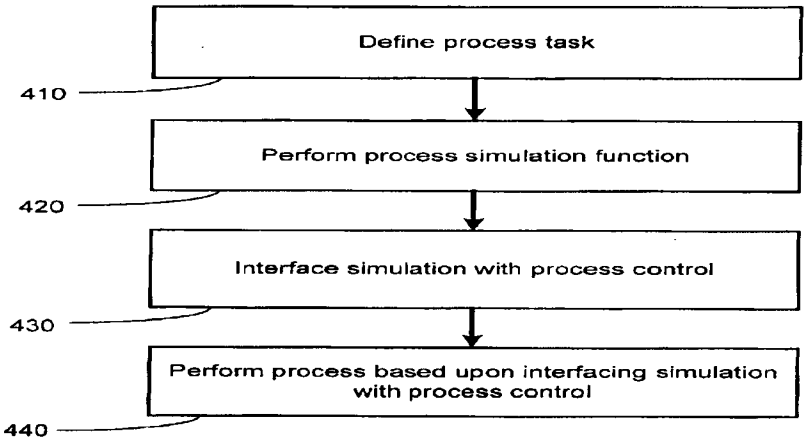



FIGURE 4

With reference to Figure 4, Sonderman et al disclose at col. 6, lines 24-47:

> Turning now to FIG. 4, a flow chart representation of the methods in accordance with the present invention is illustrated. In one embodiment, ***the system 100 defines a process task that is to be performed (block 410).*** The process task may be a photolithography process, an etching process, and the like. ***The system 100 then performs a process simulation function*** (block 420). A more detailed description of the process simulation function described in block 420, is illustrated below. In one embodiment, a simulation data set results from the execution of the process simulation function.
>
> ***Once the system 100 performs the process simulation function, the system 100 performs an interfacing function***, which facilitates interfacing of the simulation data with the process control environment 180 (block 430). The process control environment 180 can utilize the simulation data in order to modify or define manufacturing control parameters that control the actual processing steps performed by the system 100. ***Once the system 100 interfaces the simulation data with the process control environment 180, the system 100 then performs a manufacturing process*** based upon the manufacturing parameters defined by the process control environment 180 (block 440). [Emphasis added]

Hence, the process flow in Sonderman et al is straightforward:

1) define a process to be modeled,

2) model the simulation result,

3) interface simulation result to processor, and then

4) run the process under control based on the pre-existing simulation result.

In the final Office Action, the examiner disagreed with this interpretation of Sonderman et al. On page 6 of the Office Action, the examiner pointed out a part of Sonderman et al's disclosure with emphasis added by underscoring which was believed by the examiner to support his position on this matter. This characterization is repeated below for the sake of convenience with the examiner's emphasis.

> Furthermore, the simulation environment 210 can be used for feedback modification of control parameters invoked by the process control environment 180. For example, the manufacturing environment 170 can send metrology data results into the simulation environment 210. The simulation environment 210 can then use the metrology data results and perform various tests and calculations to provide more accurate, modified control parameters to the process control environment 180. A feedback loop in then completed when the process control environment 180 sends the modified or adjusted process control parameters to the manufacturing environment 170 for further processing of semiconductor wafers. {Examiner's emphasis added.}

Appellant respectfully points out that this description in Sonderman et al is a description of a **feedback loop** as Sonderman et al describe just below that portion which the examiner emphasized. Feedback modification is by definition the control of future wafers based on what has already occurred to a previous wafer. Hence, this section supports rather than refutes Appellant's position on this matter.

Accordingly, Appellant respectfully submits that Sonderman et al do not disclose and indeed ***teach away*** from the present invention where data input from an actual process being performed is used for producing a first principles simulation result, which is produced for the actual process being performed during performance of the actual process for control of the actual process.

The deficiencies in Sonderman et al are not overcome by Jain et al. The Office Action in rejecting the present claims supplements the teachings of Sonderman et al with the

teachings of Jain et al for their teaching of computer encoded differential equations in a mathematical physical engine (MPE) which can be applied to wafer processing. See Office Action, page 16. Jain et al describe at pages 372-373 that:

> We ***propose*** a wafer scale implementation of the MPE. The starting point would be a dedicated processing cell, optimized specifically for the PDE arithmetic and data routing. Because of the relative simplicity of the cell, it is expected that extremely large arrays (8x8 to 32x32) ***could be*** successfully processed on a single piece of silicon using Wafer Scale Integration techniques. In fact, we have already laid the foundation for the development of such a processing cell. Our Universal Multiply-Subtract-Add [11] ***could be*** adapted for this first cell design. Similarly, our nonlinear coprocessor cell [12]-[14] ***might be used*** in conjunction with the UMSA to provide advanced mathematical functions. As suggested in Fig. 2, there would be ***courtyards of processors***, each with two interconnection networks and two memory banks. 2-D, 3-D, and 4-D problems could then be mapped for parallel computations. Since inter-processor delays are very small (say a few ns), extremely high speeds could be achieved. This, together with the high degree of parallelism, would result also in high throughout. We ***envision*** 100 to 1000 processors (on one wafer) forming a wafer scale MPE. At a clock frequency of 50 MHz, a single wafer could achieve up to 20 GFLOPs performance. With our nonlinear coprocessor added, each instruction could equate to multiple floating point operations.
>
> Furthermore, because of the extendible architecture, several wafers ***could be*** interconnected as shown in Fig. 5 to construct a "stacked" MPE wafer system (SMPE). Note that no vertical interconnects within the stack of wafers are expected. Tens to hundreds of GFLOPs performance in a volume the size of a desk-top computer [15] ***could*** thus be achieved. However, ***these predictions*** ignore the likely technical advances in the next five years; a tenfold further increase in performance ***might be achievable***. [Emphasis Added]

Thus, as emphasized above, the proposed development work in Jain requires the development of ***futuristic*** computational equipment which one of ordinary skill in the art would be reluctant to implement or utilize for the rigorous standards needed in semiconductor manufacturing.

Appellant's position in this matter is corroborated by Tan et al, attached in the Evidence Appendix. Tan et al also teach the use on an existing process model for feedback or feed forward processing. In feedback control, by definition, the results of a process step are

provided to subsequent wafer. In feed forward control, the results of a prior process step are used to adjust a subsequent process being run of the wafer. Tan et al describe:

> The illustrative APC Framework 200 includes a process model 202 that receives ***feed-forward and feed-back data*** and calculates a processing parameter. The illustrative portion of the APC Framework 200 includes two measurement devices, in particular a pre-process metrology machine 204 and a post-processing metrology machine 206. The pre-process metrology machine 204 performs a measurement on a material prior to processing in a processing machine 208 and sends the measurement, as feed-forward data, to the process model 202. The processing machine 208 sends processed material to the post-processing metrology machine 206 ***to measure post-process data which is sent to the process model 202 as feedback data.***
>
> Referring to FIG. 4, a schematic block diagram shows material flow of a processing step 400 of a semiconductor manufacturing process from a process engineer perspective. An APC plan 402 retrieves a process model from the data store 306, then executes a parameter calculation algorithm 404. The APC plan 402 gives the calculated parameters to a machine 406 and directs the machine 406 to execute the process. The machine 406 issues a signal to the APC plan 402 ***when the process execution is complete.*** The APC plan 402 sends the calculated parameters to the data history store 310 of the historical database 312.
>
> Referring to FIG. 5, a schematic block diagram shows material flow of a post-process measurement step 500 of a semiconductor manufacturing process from a process engineer perspective. An APC plan 502 sends a message to a machine 504 instructing the machine 504 to measure a post-processed material. The machine 504 sends measurement data to the APC plan 502. The APC plan 502 retrieves an old process model from the data store 306. The APC plan 502 executes a model update algorithm 506. The APC plan 502 ***stores an updated model in the data store 306 for usage in the processing step 400.*** The APC plan 502 sends new model data to the data history store 310 of the historical database 312. [Emphasis added.]

Thus, Tan et al use post-process data to update a model for a subsequent process step.

Appellant's position on these matters is also supported by Kee et al, of record in this application and attached in the Evidence Appendix. In the outstanding Office Action, the examiner indicated that he found no connection or legal basis for considering the teachings of Kee et al. Recently published guidelines for the Patent and Trademark Office, published in Federal Register Vol. 72, No. 195, on Wednesday October 10, 2007entitled: "Examination

Guidelines for Determining Obviousness under 35 U.S.C. 103 in View of the Supreme Court Decision in KSR International v. Teleflex Inc," indicate that:

> Office personnel should consider all rebuttal evidence that is timely presented by the Applicants when reevaluating any obviousness determination. Rebuttal evidence may include evidence of "secondary considerations," such as "commercial success, long felt but unsolved needs, [and] failure of others", and may also include evidence of unexpected results. Office personnel must articulate findings of fact that support the rationale relied upon in an obviousness rejection. As a result, Applicants are likely to submit evidence to rebut the fact finding made by Office personnel. For example, in the case of a claim to a combination, Applicants may submit evidence or argument to demonstrate that:
>
> (1) one of ordinary skill in the art could not have combined the claimed elements by known methods (e.g., ***due to technological difficulties***);
>
> (2) the elements in combination do not merely perform the function that each element performs separately; or
>
> (3) the results of the claimed combination were ***unexpected***.
>
> Once the Applicant has presented rebuttal evidence, Office personnel should reconsider any initial obviousness determination in view of the entire record. All the rejections of record and proposed rejections and their bases should be reviewed to confirm the continued viability. The Office action should clearly communicate the Office's findings and conclusions, articulating how the conclusions are supported by the findings. [Emphasis Added.]

M.P.E.P. § 2143.01(II) indicates that all teachings in the prior art must be considered. M.P.E.P. 2141.03 indicates that the examiner must ascertain what would have been obvious to one of ordinary skill in the art at the time of the invention. Hence, for all these legal considerations, Kee et al is presented as rebuttal evidence for the non-obviousness of the claims.

Specifically, the prior art Kee et al reference is evidence of the **technological difficulties** involved in producing a first principles model simulation result and, under the published guidelines, should be considered. The prior art Kee et al reference represents what one of ordinary skill in the art would have known and would have expected at the time of the invention and, under the M.P.E.P., should be considered.

Kee et al deal with the process control of a Rapid Thermal Processing (RTP) tool and do **not** use real time modeling. RTP tools are tools used in semiconductor manufacturing. Kee et al in detail disclose that:

> The modeling apparatus 101 of the instant invention may also be used to perform an inverse analysis to establish the boundary conditions or parameter values required to achieve a certain function of the thermal system. This allows the apparatus to be used to establish the appropriate process parameters and boundary conditions for the thermal system modeled. In accordance with the instant invention, the inverse analysis can be directly carried out by the modeling apparatus ***rather than using the conventional approach, which merely solves the direct problem repeatedly, in a lengthy and costly iterative process***, to determine appropriate input parameters to achieve a desired result. In other words, in accordance with the instant invention, ***once a particular thermal process is modeled for a particular set of control parameters***, the device may then be used to automatically obtain the necessary control parameters to achieve a desired result by providing the modeling apparatus with parameters corresponding to the desired result.
>
> To carry out the inverse analysis, the modeling apparatus 101 includes an inverse parameter input section 104 also connected to input device 103. A user inputs into the modeling apparatus 101 parameters corresponding to desired results, e.g., desired temperature characteristics of the system, which are stored in memory 108. The processing unit 110, under control of modeling program 111, ***uses the previously generated model*** of the thermal system and the parameters held in memory 108 and derives or predicts particular control parameters to meet the constraints entered through the inverse parameter input section 104. This process is more fully described below in connection with the examples provided.[2] [Emphasis added.]

Hence, Kee et al explicitly disclose that a ***previously generated*** model of the thermal system is used to design and control the thermal system. Kee et al exemplify the difficulties of a "conventional approach" which solves spectral radiation transport equations through "a lengthy and costly iterative process." These problems forced Kee et al to use ***pre-generated model results*** for a control process of a RTP process.

The examiner in the final Office Action did not apply but rather noted the IEEE 1990 paper by Su-shing Chen, "AEMPES: An expert system for in-situ diagnostics and process

[2] Kee et al, col. 4, lines 21-50.

monitoring," hereinafter referred to as AEMPS, as evidence that the most recently added limitation (said first principles simulation result being produced in a time frame shorter in time than the actual process being performed) is known in the art. Yet, AEMPS describes the use of simulation in neural network environment used to "learn processes and the equipment model." See page 120, section 4. AEMPS describes in section 4 that "a rule-based expert system provides human interfaces and high-level decision support." Accordingly, AEMPS does **not** describe a first principles simulation result, but rather describes a neural network learning-based simulation. Accordingly, a system such as in AEMPS which learns a behavior and establishes rules based on the behavior would be used in a feedback control (see section 4 of AEMPS). Such a system would **not** 1) produce a first principles simulation result or 2) produce a first principles simulation result during the performance of the actual process to control the actual process performed by the semiconductor processing tool.

Moreover, AEMPS describes in section 2 (regarding manufacturing automation) that it is **not known** how to couple computer aided design, the integration of a manufacturing line, and its simulator together, and that it is **not known** how "to complete integration of manufacturing lines with their simulators." Hence, like Jain et al, AEMPS describes a ***futuristic*** system under development. Even if for the sake of argument it were supposed that the AEMPS system was a first principles simulation result (which it is not), one of ordinary skill in the art would be reluctant to implement or utilize the AEMPS system for the rigorous standards needed in semiconductor manufacturing.

The examiner in the final Office Action also noted appellant's disclosure at numbered paragraphs [0004] and [0005] in the specification as an apparent admission that the feature of a first principles simulation result being produced in a time frame shorter in time than the actual process being performed was a feature known in the art. Yet, the specification

describes this material as being background material and makes no indication that what the present inventors recognized and described in the background section was known to others or would in any other way qualify as 35 U.S.C. § 102 prior art.

More importantly, numbered paragraphs [0004] and [0005] indicate at most that the times for a large number of simulations ***typically done in the tool design stage*** are comparable to wafer or wafer cassette processing times. There is no statement here regarding how long the times would be for a process control simulation. Further, numbered paragraphs [0004] and [005] indicate that, at the time of the invention, there were serious impediments which would mean that it would not be possible, prior to the invention, to produce a first principles simulation result in a time frame shorter in time than the actual process being performed.

> **[0004]** These industry and manufacturing challenges have led to interest in more use of computer based modeling and simulation in the semiconductor manufacturing industry. Computer-based modeling and simulation are increasingly being used for prediction of tool performance during the semiconductor manufacturing tool design process. The use of modeling allows the reduction of both cost and time involved in the tool development cycle. Modeling in many disciplines, such as stress, thermal, magnetics, etc., has reached a level of maturity where it can be trusted to provide accurate answers to design questions. Moreover, computer power has been increasing rapidly along with the development of new solution algorithms, both of which resulted in reduction of time required to obtain a simulation result. ***Indeed, the present inventors have recognized that a large number of simulations typically done in the tool design stage can presently be run in times comparable to wafer or wafer cassette processing times.*** These trends have led to the suggestion that simulation capability typically used only for tool design can be implemented directly on the tool itself to aid in various processes performed by the tool. For example, the 2001 International Technology Roadmap for Semiconductors ***identifies issues impeding the development of on-tool integrated simulation capability*** as an enabling technology for manufacturing very small features in future semiconductor devices.
>
> **[0005]** Indeed, ***the failure of industry to implement on-tool simulation to facilitate tool processes is primarily due to the need for computational resources capable of performing the simulations in a reasonable time.*** Specifically, the processor capabilities currently dedicated to semiconductor

> manufacturing tools are typically limited to diagnostic and control functions, and therefore could only perform relatively simple simulations. Thus, the semiconductor manufacturing industry has perceived ***a need to provide powerful dedicated computers in order to realize meaningful on-tool simulation capabilities.*** However, dedication of such a computer to the semiconductor processing tool results in wasted computational resources when the tool runs processes that use simple simulations, or no simulations at all. This inefficient use of an expensive computational resource has been ***a major impediment to implementation of simulation capabilities on semiconductor processing tools.*** [Emphasis added.]

Hence, Tan et al, Jain et al, Kee et al, AEMPES, and the background section of the specification **all** discredit any suggestion that the examiner may have read from the disclosure of Sonderman et al for real-time simulation and control of an actual process being performed.

The Supreme Court in *KSR International Co. v. Teleflex Inc. et al. 2007 U.S. LEXIS 4745* reinforced the role of *Graham* factors, teaching away and elements working together in an unexpected and fruitful manner in deciding obviousness. The Court stated that:

> In *United States v. Adams*, 383 U. S. 39, 40 (1966), a companion case to *Graham*, the Court considered the obviousness of a wet battery that varied from prior designs in two ways: It contained water, rather than the acids conventionally employed in storage batteries; and its electrodes were magnesium and cuprous chloride, rather than zinc and silver chloride. The Court recognized that when a patent claims a structure already known in the prior art that is altered by the mere substitution of one element for another known in the field, the combination must do more than yield a predictable result. 383 U. S., at 50-51. It nevertheless rejected the Government's claim that Adams's battery was obvious. The Court relied upon the corollary principle that when the prior art ***teaches away*** from combining certain known elements, discovery of a successful means of combining them is more likely to be nonobvious. *Id.*, at 51-52. When Adams designed his battery, the prior art warned that risks were involved in using the types of electrodes he employed. The fact that the elements worked together in ***an unexpected and fruitful manner*** supported the conclusion that Adams's design was ***not obvious*** to those skilled in the art. [Emphasis added.]

In the present situation, the claimed elements worked together in ***an unexpected and fruitful manner*** as compared to the prior art. For example, since in Sonderman et al there are ***new control inputs*** for each subsequent wafer, one can not compensate for real time excursions from the existing model occurring while the wafer is being processed. In other

words, the historically lengthy time for generation of a first principles model simulation would mean that, in Sonderman et al, one is prevented from realizing a real time process control based on a first principles simulation during the actual process being performed. Meanwhile, the claimed processes and systems (by producing a first principles simulation result in a time frame shorter in time than the actual process being performed) permits accurate control of the process even if the system being controlled deviates from its historical behavior.

For all these reasons, Appellant submits that independent Claims 1, 23, and 48 patentably define over Sonderman et al and Jain et al and Tan et al.

Hence, the 35 U.S.C. § 103(a) rejection of Claims 1-44 and 48 as being unpatentable over Sonderman et al in view of Jain et al should be reversed.

### C. Regarding the 35 USC 103 Rejection of Claims 49 and 50 over Sonderman et al and Jain et al

Claim 49 defines that the performing a first principles simulation includes providing for the first principles simulation a reuse of known solutions as initial conditions for the first principles simulation. The Office Action notes that "Jain teaches use of Navier Stokes and other known simulation solutions" and cites pp. 367-368 of Jain et al. However, the Navier Stokes equation on page 367 of Jain et al is a fluid flow equation which needs boundary conditions and which need s to be solved in order to produce a solution. The Navier Stokes equation on page 367 of Jain et al does **not** represent a solution, much less the reuse of known solutions as initial conditions for the first principles simulation. Appellant's inspection of the remainder of Jain et al finds no disclosure of the reuse of known solutions as initial conditions for the first principles simulation.

Hence, for this additional reason (besides their dependence from allowable claims),

the 35 U.S.C. § 103(a) rejection of Claims 49 and 50 as being unpatentable over Sonderman et al in view of Jain et al should be reversed.

**D. Regarding the Double Patenting Rejections**

**1. The Double Patenting Rejection over the '583 Application**

The filed terminal disclaimer addressed this issue. Hence, Appellant has overcome the double patenting rejection.

**2. The Double Patenting Rejection over the '138 Application**

The filed terminal disclaimer addressed this issue. Hence, Appellant has overcome the double patenting rejection.

**3. The Double Patenting Rejection over the '507 Application**

The filed terminal disclaimer addressed this issue. Hence, Appellant has overcome the double patenting rejection.

## VII. 41.37(c)(1)(vii) Claims Appendix Of Claims Involved In Appeal

Attached herewith is a Claims Appendix.

## IX. 41.37(C)(1)(IX) Evidence Appendix

Included in the appendix is a copy of Tan et al (U.S. Pat. No. 6,263,255).

Included in the appendix is a copy of Kee et al (U.S. Pat. No. 5,583,780).

Included in the appendix is a copy of Su-shing Chen, "AEMPES: An expert system for in-situ diagnostics and process monitoring," referred to by the examiner, but not applied, in the Office Action of February 7, 2008.

## X. 41.37(c)(1)(x) Related Proceedings Appendix

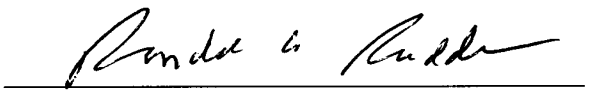
There are no related proceedings.

## XI. Conclusion

Appellant request on the basis of the arguments presented above that the outstanding grounds for the rejection be reversed. Appellant submits that the application is in condition for allowance.

Respectfully submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.



Ronald A. Rudder, Ph. D.
Registration No. 45,618
Attorney of Record
OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.
1940 Duke Street
Alexandria, Virginia 22314
(703) 412- 7033(Direct Dial)
(703) 413-2220 (Facsimile)
RRUDDER@OBLON.COM

Customer Number
22850

Tel: (703) 413-3000
Fax: (703) 413 -2220
(OSMMN 06/04)

I:\ATTY\RAR\AMENDMENTS\230421US\APPEAL.DOC

## CLAIMS APPENDIX

1. A method of facilitating a process performed by a semiconductor processing tool, comprising:

inputting process data relating to an actual process being performed by the semiconductor processing tool;

inputting a first principles physical model including a set of computer-encoded differential equations, the first principles physical model describing at least one of a basic physical or chemical attribute of the semiconductor processing tool;

performing first principles simulation for the actual process being performed using the physical model to provide a first principles simulation result in accordance with the process data relating to the actual process being performed in order to simulate the actual process being performed, said first principles simulation result being produced in a time frame shorter in time than the actual process being performed; and

using the simulation result as part of a data set that characterizes the actual process being performed by the semiconductor processing tool.

2. The method of Claim 1, wherein said inputting process data comprises directly inputting the data relating to the actual process being performed by the semiconductor processing tool from at least one of a physical sensor and a metrology tool physically mounted on the semiconductor processing tool.

3. The method of Claim 1, wherein said inputting process data comprises indirectly inputting the data relating to the actual process being performed by the semiconductor processing tool from at least one of a manual input device and a database.

4. The method of Claim 3, wherein said indirectly inputting comprises inputting data recorded from a process previously performed by the semiconductor processing tool.

5. The method of Claim 3, wherein said indirectly inputting comprises inputting data set by a simulation operator.

6. The method of Claim 1, wherein said inputting process data comprises inputting the data relating to a process performed by the semiconductor processing tool as virtual sensor data from a simulation module.

7. The method of Claim 1, wherein said inputting process data comprises inputting data relating to at least one of the physical characteristics of the semiconductor processing tool and the semiconductor tool environment.

8. The method of Claim 1, wherein said inputting data comprises inputting data relating to at least one of a characteristic and a result of a process performed by the semiconductor processing tool.

9. The method of Claim 1, wherein said inputting a first principles physical model comprises inputting a spatially resolved model of the geometry of the semiconductor processing tool.

10. The method of Claim 1, wherein said inputting a first principles physical model comprises inputting fundamental equations necessary to perform first principles simulation to obtain a simulation result that can form part of a data set that characterizes the process performed by the semiconductor processing tool.

11. The method of Claim 1, wherein said performing first principles simulation comprises performing first principles simulation concurrently with the process performed by the semiconductor processing tool.

12. The method of Claim 11, wherein said performing first principles simulation comprises performing first principles simulation to provide a simulation result that is a variation of a parameter tested by the concurrent process performed by the semiconductor processing tool.

13. The method of Claim 11, wherein said performing first principles simulation comprises performing first principles simulation to provide a simulation result relating to a different parameter than a parameter tested by the concurrent process performed by the semiconductor processing tool.

14. The method of Claim 1, further comprising performing first principles simulation not concurrently with the process performed by the semiconductor processing tool.

15. The method of Claim 1, further comprising storing the data set in a library for subsequent use processes performed by the semiconductor processing tool.

16. The method of Claim 1, further comprising using a network of interconnected resources to perform at least one of the process steps recited in Claim 1.

17. The method of Claim 16, further comprising using code parallelization among interconnected computational resources to share the computational load of the first principles simulation.

18. The method of Claim 16, further comprising sharing simulation information among interconnected resources to facilitate a process performed by the semiconductor processing tool.

19. The method of Claim 18, wherein said sharing simulation information comprises distributing simulation results among the interconnected resources to reduce redundant execution of substantially similar first principles simulations by different resources.

20. The method of Claim 18, wherein said sharing simulation information comprises distributing model changes among the interconnected resources to reduce redundant refinements of first principles simulations by different resources.

21. The method of Claim 18, further comprising using remote resources via a wide area network to facilitate the semiconductor process performed by the semiconductor processing tool.

22. The method of Claim 21, wherein said using remote resources comprises using at least one of remote computational and storage resources via a wide area network to facilitate the semiconductor process performed by the semiconductor processing tool.

23. A system comprising:

a semiconductor processing tool configured to perform an actual process;

an input device configured to input process data relating to the actual process being performed by the semiconductor processing tool; and

a first principles simulation processor configured to:

input a first principles physical model including a set of computer-encoded differential equations describing at least one of a basic physical or chemical attribute the semiconductor processing tool, and

perform first principles simulation for the actual process being performed using the physical model to provide a first principles simulation result in accordance with the process data relating to the actual process being performed in order to simulate the actual process being performed, said first principles simulation result being produced in a time frame shorter in time than the actual process being performed, wherein the simulation result is used as part of a data set that characterizes the process performed by the semiconductor processing tool.

24. The system of Claim 23, wherein said input device comprises at least one of a physical sensor and a metrology tool physically mounted on the semiconductor processing tool.

25. The system of Claim 23, wherein said input device comprises at least one of a manual input device and a database.

26. The system of Claim 25, wherein said input device is configured to input data recorded from a process previously performed by the semiconductor processing tool.

27. The system of Claim 25, wherein said input device is configured to input data set by a simulation operator.

28. The system of Claim 23, wherein said input device is configured to input the data relating to a process performed by the semiconductor processing tool as virtual sensor data from a simulation module.

29. The system of Claim 23, wherein said input device is configured to input data relating to at least one of the physical characteristics of the semiconductor processing tool and the semiconductor tool environment.

30. The system of Claim 23, wherein said input device is configured to input data relating to at least one of a characteristic and a result of a process performed by the semiconductor processing tool.

31. The system of Claim 23, wherein said processor is configured to input a first principles physical model comprising a spatially resolved model of the geometry of the semiconductor processing tool.

32. The system of Claim 23, wherein said processor is configured to input a first principles physical model comprising fundamental equations necessary to perform first principles simulation to obtain a simulation result that can form part of a data set that characterizes the process performed by the semiconductor processing tool.

33. The system of Claim 23, wherein said processor is configured to perform said first principles simulation concurrently with the process performed by the semiconductor processing tool.

34. The system of Claim 33, wherein said processor is configured to perform the first principles simulation to provide a simulation result that is a variation of a parameter tested by the concurrent process performed by the semiconductor processing tool.

35. The system of Claim 33, wherein said processor is configured to perform the first principles simulation to provide a simulation result relating to a different parameter than a parameter tested by the concurrent process performed by the semiconductor processing tool.

36. The system of Claim 23, wherein said processor is configured to perform said first principles simulation not concurrently with the process performed by the semiconductor processing tool.

37. The system of Claim 23, wherein said processor is further configured to store the data set in a library for subsequent use processes performed by the semiconductor processing tool.

38. The system of Claim 23, further comprising a network of interconnected resources connected to said processor and configured to assist said processor in performing at least one of the inputting a first principles simulation model and performing a first principles simulation.

39. The system of Claim 38, wherein said network of interconnected resources is configured to use code parallelization with said processor to share the computational load of the first principles simulation.

40. The system of Claim 38, wherein said network of interconnected resources is configured to share simulation information with said processor to facilitate said process performed by the semiconductor processing tool.

41. The system of Claim 40, wherein said network of interconnected resources is configured to distribute simulation results to said processor to reduce redundant execution of substantially similar first principles simulations.

42. The system of Claim 40, wherein said network of interconnected resources is configured to distribute model changes to said processor to reduce redundant refinements of first principles simulations.

43. The system of Claim 38, further comprising remote resources connected to said processor via a wide area network and configured to facilitate the semiconductor process performed by the semiconductor processing tool.

44. The system of Claim 43, wherein said remote resources comprise at least one of a computational and a storage resource.

48. At least one of non-volatile media and volatile media containing program instructions for execution on a processor, which when executed by the computer system, cause the processor to perform the steps of:

inputting process data relating to an actual process being performed by the semiconductor processing tool;

inputting a first principles physical model including a set of computer-encoded differential equations, the first principles physical model describing at least one of a basic physical or chemical attribute of the semiconductor processing tool;

performing first principles simulation for the actual process being performed using the physical model to provide a first principles simulation result in accordance with the process data relating to the actual process being performed in order to simulate the actual process being performed, said first principles simulation result being produced in a time frame shorter in time than the actual process being performed; and

using the simulation result as part of a data set that characterizes the actual process being performed by the semiconductor processing tool.

49. The method of Claim 1, wherein said performing a first principles simulation comprises:

providing for the first principles simulation a reuse of known solutions as initial conditions for the first principles simulation.

50. The system of Claim 23, wherein the first principles simulator is configured to provide for the first principles simulation a reuse of known solutions as initial conditions for the first principles simulation.

# EVIDENCE APPENDIX

1. Tan et al (U.S. Pat. No. 6,263,255)


US006263255B1

(12) **United States Patent**
**Tan et al.**

(10) **Patent No.: US 6,263,255 B1**
(45) **Date of Patent: Jul. 17, 2001**

(54) **ADVANCED PROCESS CONTROL FOR SEMICONDUCTOR MANUFACTURING**

(75) Inventors: **Heng-Wei Osbert Tan; Donald H. Vines, Jr.**, both of Austin, TX (US)

(73) Assignee: **Advanced Micro Devices, Inc.**, Sunnyvale, CA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/080,620**

(22) Filed: **May 18, 1998**

(51) **Int. Cl.**[7] ..... **G06F 19/00**

(52) **U.S. Cl.** ..... **700/121**; 700/48; 700/95; 700/106; 700/115; 700/116; 709/223; 709/303; 709/304; 705/14; 705/15; 705/27; 705/28

(58) **Field of Search** ..... 700/48, 49, 95, 700/96, 97, 106, 115, 116, 121; 709/223-226, 300, 303, 304; 705/14, 15, 27, 20, 8

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,552,718 * 11/1985 Impink, Jr. et al. ..... 376/216
5,859,964 * 1/1999 Wang et al. ..... 714/48
5,896,294 * 4/1999 Chow et al. ..... 700/121
6,038,540 * 3/1999 Krist et al. ..... 705/8
6,038,545 * 3/1999 Mandeberg et al. ..... 705/15

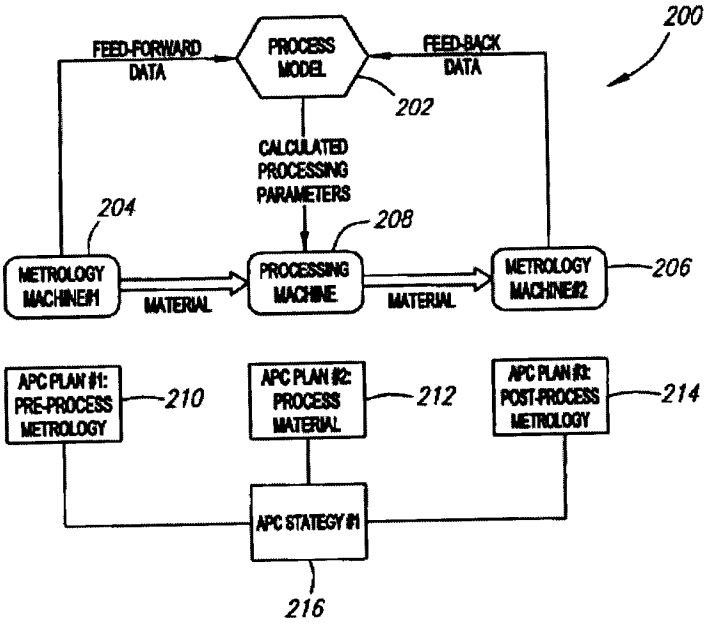
* cited by examiner

*Primary Examiner*—William Grant
*Assistant Examiner*—Ramesh Patel
(74) *Attorney, Agent, or Firm*—Skjerven Morrill MacPherson LLP; Ken J. Koestner

(57) **ABSTRACT**

An Advanced Process Control (APC) Framework performs automatic process control operations through the design and development of a software framework that integrates factory, process, and equipment control systems. The APC Framework benefits semiconductor-manufacturing factories, or "fabs," throughout the development of the APC Framework by using an iterative development approach. The APC Framework is designed to integrate seamlessly with commercially-available APC tools. The APC Framework specifies components and a component structure that enable multiple vendors to build and sell framework-compatible products using an open architecture that accommodates plug-and-play components. The APC Framework advantageously increases product yield distributions and equipment utilization, and lowers defect densities.

**37 Claims, 22 Drawing Sheets**



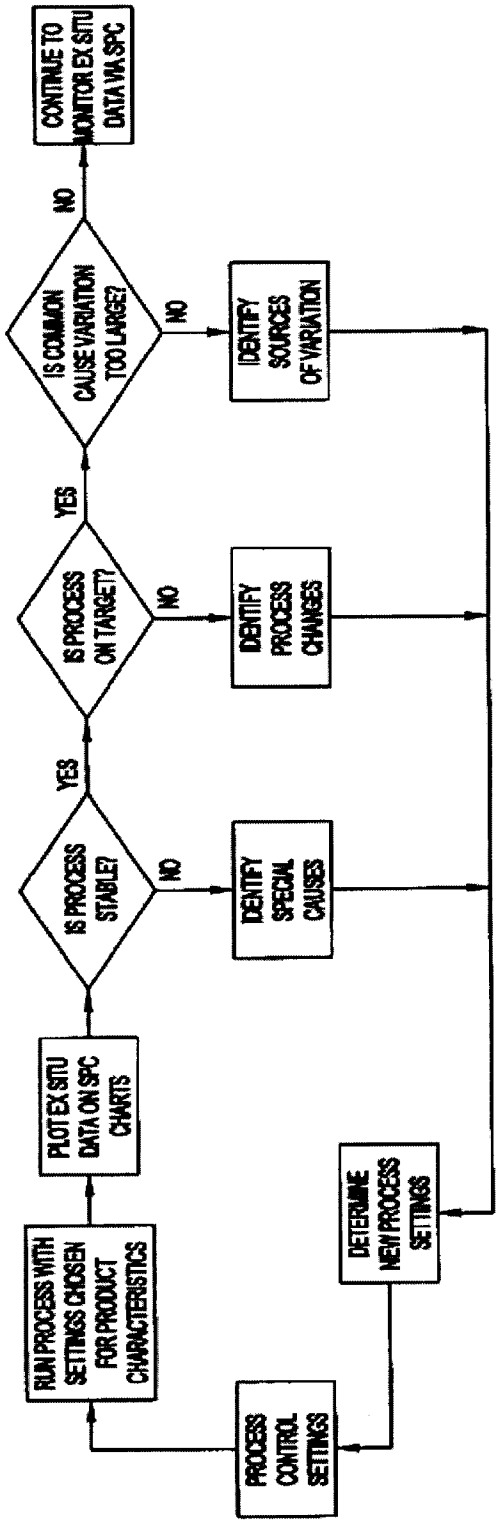



*FIG. 1*
*(PRIOR ART)*

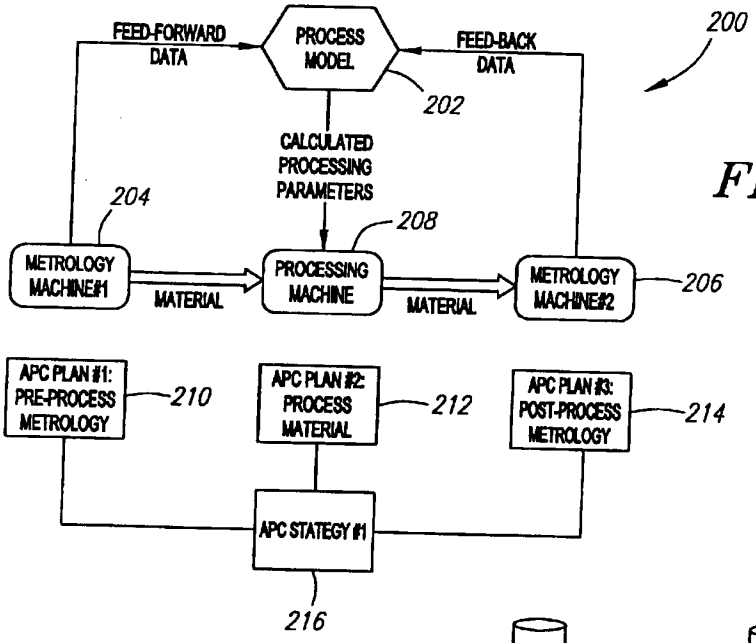

FEED-FORWARD DATA
PROCESS MODEL
FEED-BACK DATA
200
202
CALCULATED PROCESSING PARAMETERS
204
208
METROLOGY MACHINE#1
MATERIAL
PROCESSING MACHINE
MATERIAL
METROLOGY MACHINE#2
206
APC PLAN #1: PRE-PROCESS METROLOGY
210
APC PLAN #2: PROCESS MATERIAL
212
APC PLAN #3: POST-PROCESS METROLOGY
214
APC STATEGY #1
216

FIG. 2

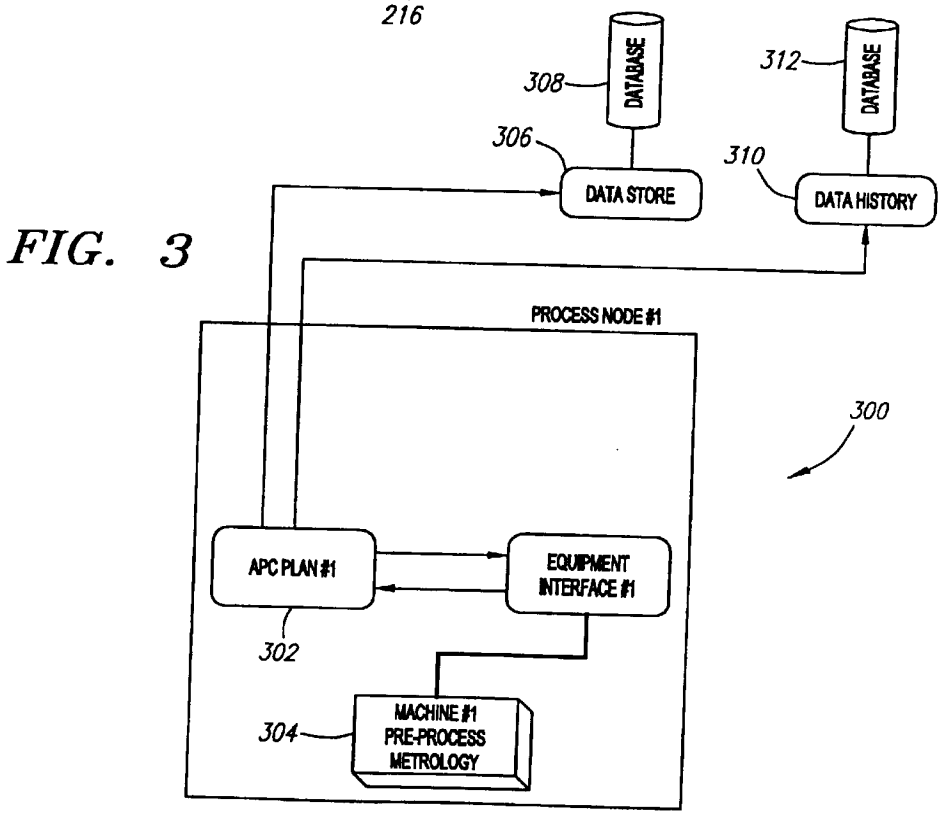

308
DATABASE
312
DATABASE
306
310
DATA STORE
DATA HISTORY
PROCESS NODE #1
300
APC PLAN #1
EQUIPMENT INTERFACE #1
302
304
MACHINE #1 PRE-PROCESS METROLOGY

FIG. 3

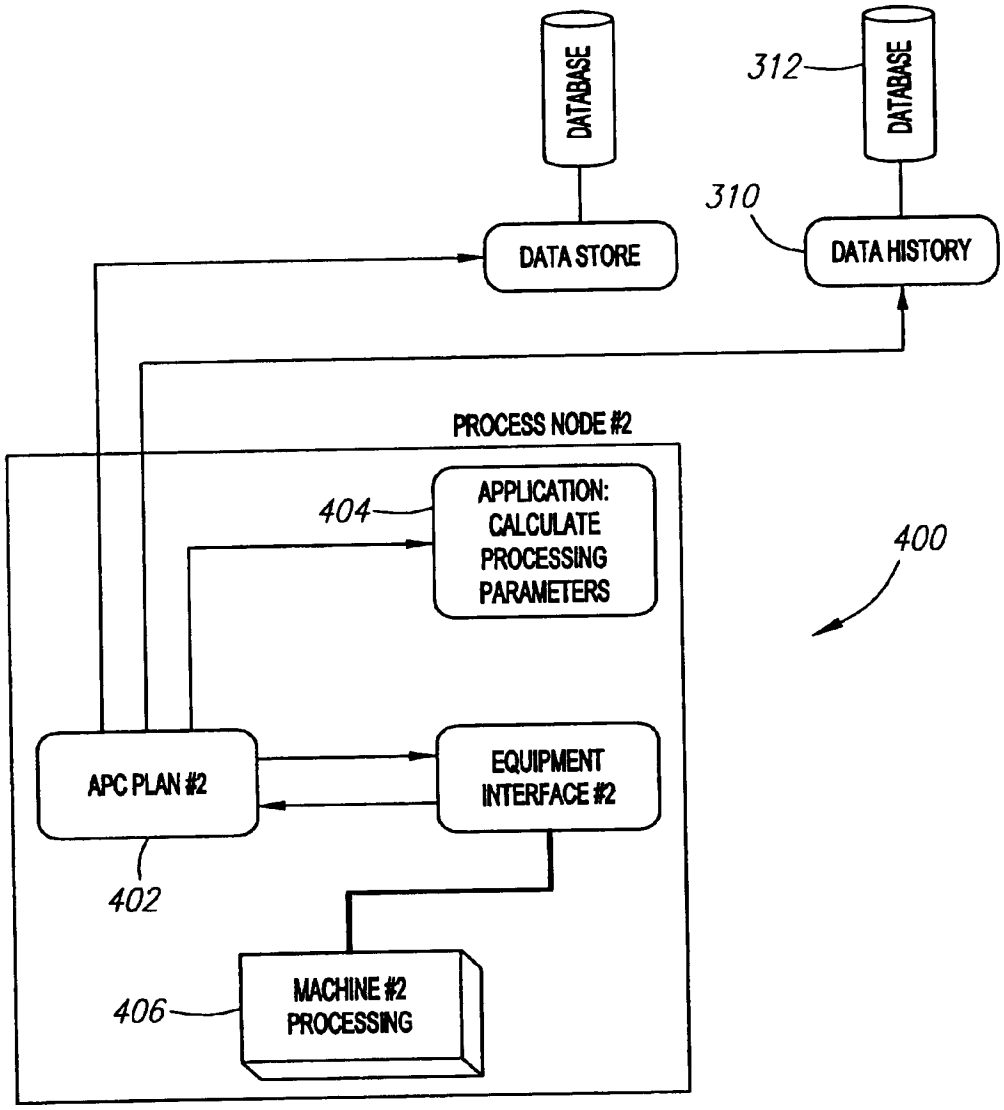

DATABASE
312
DATABASE
310
DATA STORE
DATA HISTORY
PROCESS NODE #2
404
APPLICATION: CALCULATE PROCESSING PARAMETERS
400
APC PLAN #2
EQUIPMENT INTERFACE #2
402
406
MACHINE #2 PROCESSING

*FIG. 4*

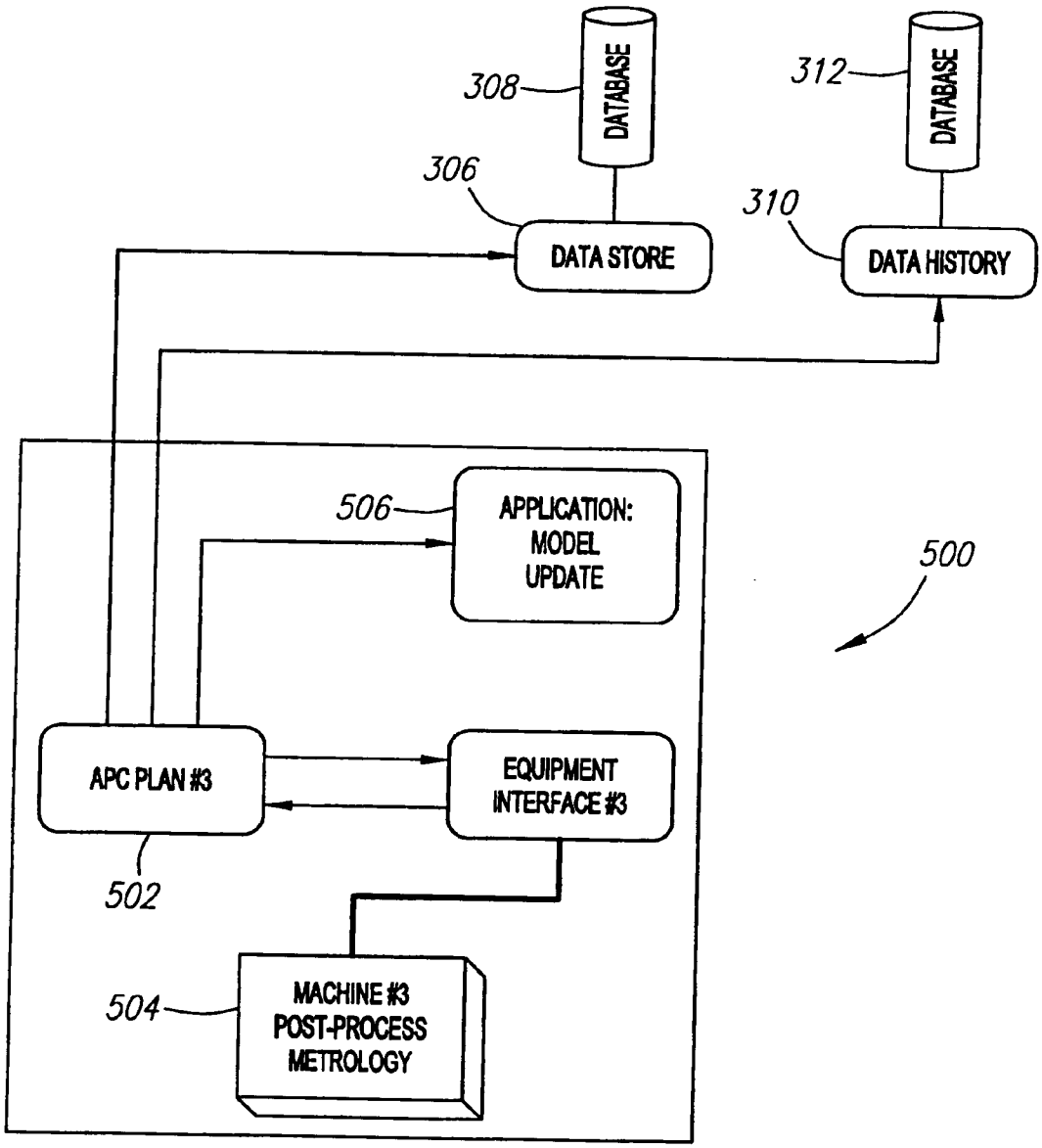

308
DATABASE
312
DATABASE
306
310
DATA STORE
DATA HISTORY
506
APPLICATION:
MODEL
UPDATE
500
APC PLAN #3
EQUIPMENT
INTERFACE #3
502
504
MACHINE #3
POST-PROCESS
METROLOGY

*FIG. 5*

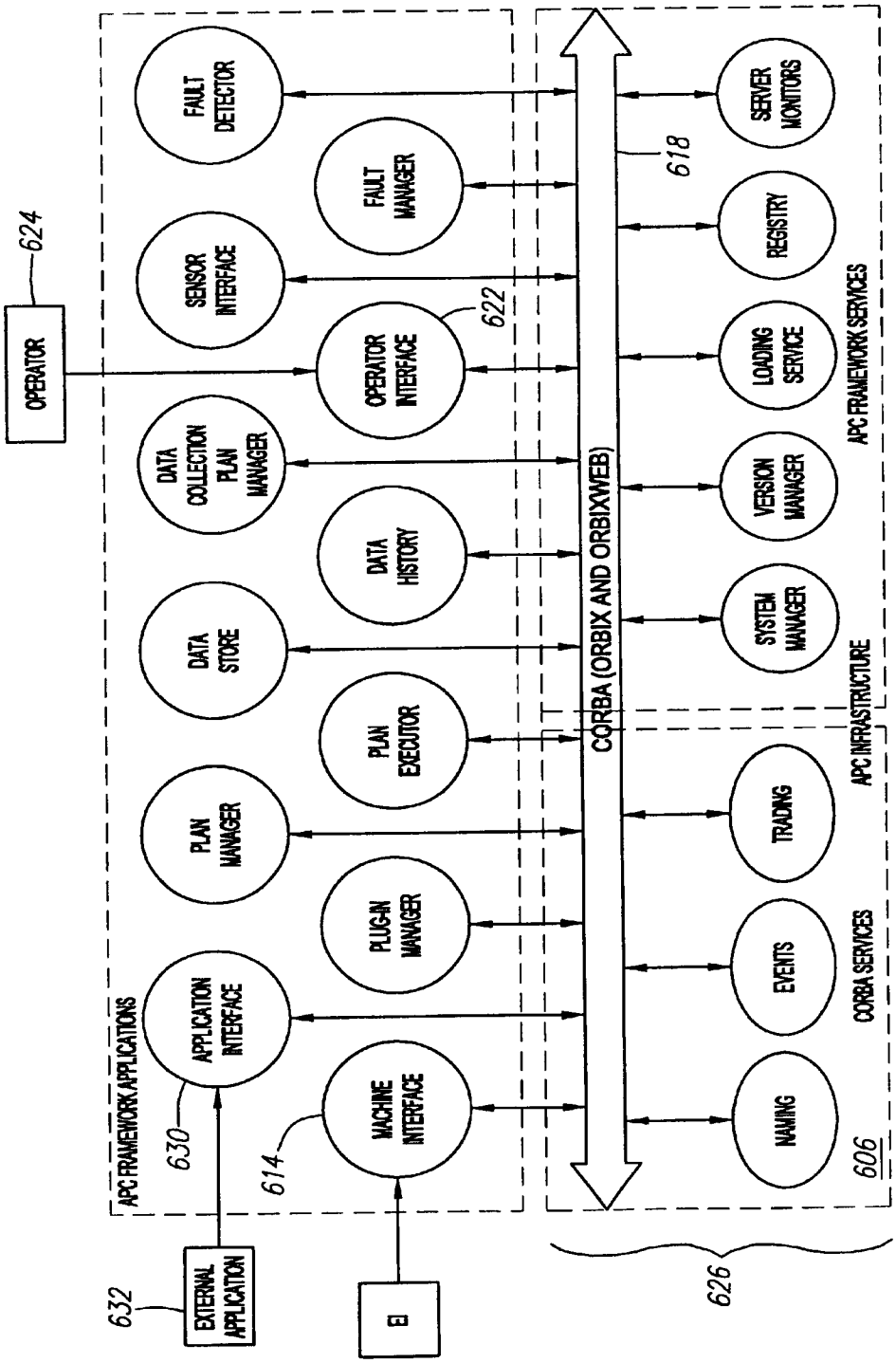

624
OPERATOR
632
EXTERNAL APPLICATION
E
APC FRAMEWORK APPLICATIONS
630
APPLICATION INTERFACE
614
MACHINE INTERFACE
PLUG-IN MANAGER
PLAN MANAGER
PLAN EXECUTOR
DATA STORE
DATA HISTORY
DATA COLLECTION PLAN MANAGER
OPERATOR INTERFACE
622
SENSOR INTERFACE
FAULT MANAGER
FAULT DETECTOR
CORBA (ORBIX AND ORBIXWEB)
618
SERVER MONITORS
REGISTRY
LOADING SERVICE
VERSION MANAGER
SYSTEM MANAGER
APC FRAMEWORK SERVICES
APC INFRASTRUCTURE
TRADING
EVENTS
NAMING
CORBA SERVICES
606
626

FIG. 6

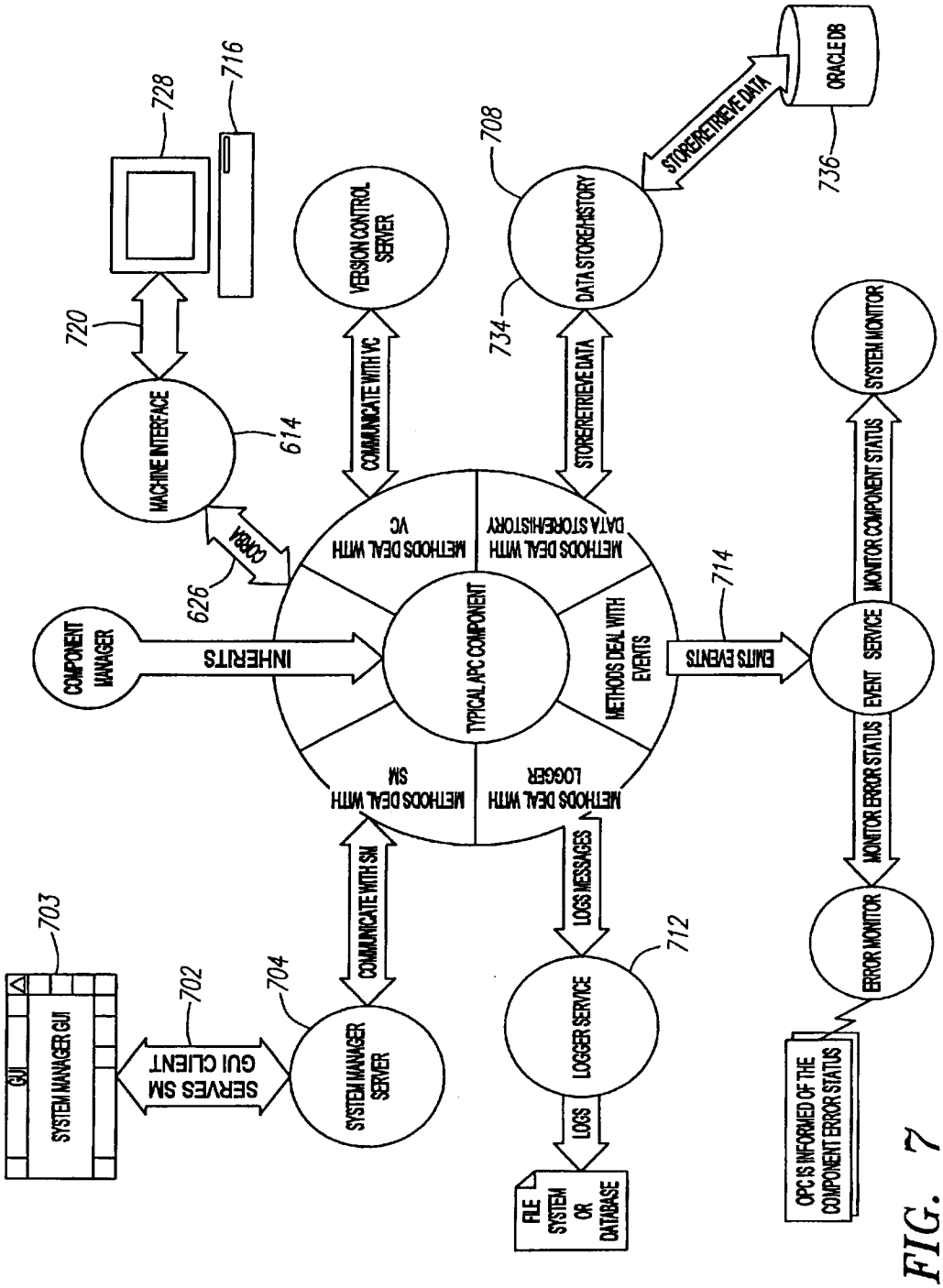

716
728
720
MACHINE INTERFACE
614
626
CORBA
VERSION CONTROL SERVER
COMMUNICATE WITH VC
708
DATA STORE/HISTORY
734
STORE/RETRIEVE DATA
STORE/RETRIEVE DATA
ORACLE DB
736
SYSTEM MONITOR
MONITOR COMPONENT STATUS
COMPONENT MANAGER
INHERITS
METHODS DEAL WITH VC
METHODS DEAL WITH DATA STORE/HISTORY
TYPICAL APC COMPONENT
METHODS DEAL WITH EVENTS
METHODS DEAL WITH SM
METHODS DEAL WITH LOGGER
714
EMITS EVENTS
EVENT SERVICE
MONITOR ERROR STATUS
703
GUI
SYSTEM MANAGER GUI
702
SERVES SM GUI CLIENT
704
SYSTEM MANAGER SERVER
COMMUNICATE WITH SM
LOGS MESSAGES
712
LOGGER SERVICE
LOGS
FILE SYSTEM OR DATABASE
ERROR MONITOR
OPC IS INFORMED OF THE COMPONENT ERROR STATUS

FIG. 7

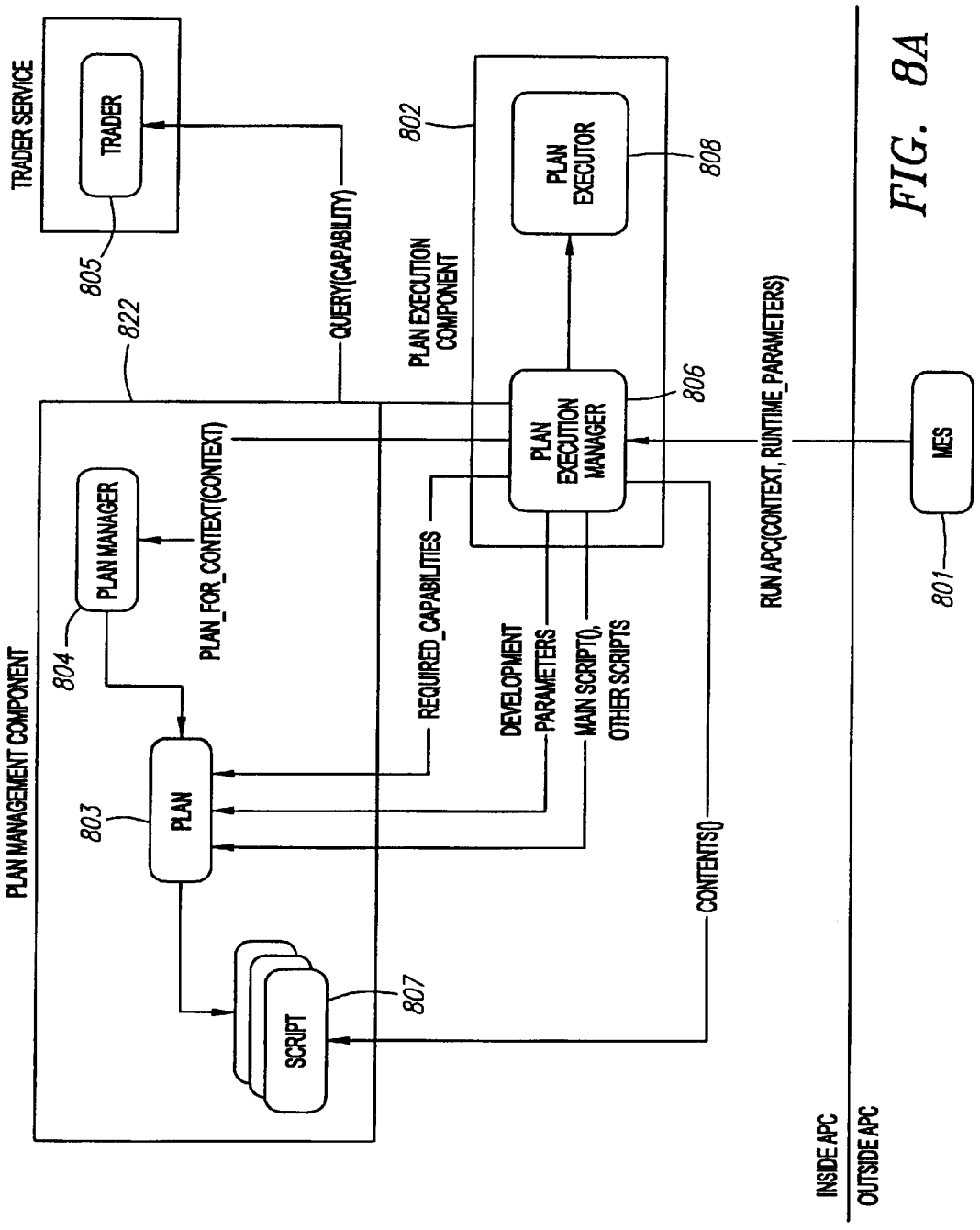

TRADER SERVICE
TRADER
805
822
QUERY(CAPABILITY)
802
PLAN EXECUTOR
808
PLAN EXECUTION COMPONENT
PLAN EXECUTION MANAGER
806
RUN APC(CONTEXT, RUNTIME_PARAMETERS)
MES
801
PLAN MANAGEMENT COMPONENT
PLAN MANAGER
804
PLAN_FOR_CONTEXT(CONTEXT)
REQUIRED_CAPABILITIES
DEVELOPMENT PARAMETERS
MAIN SCRIPT(), OTHER SCRIPTS
PLAN
803
CONTENTS()
SCRIPT
807
INSIDE APC
OUTSIDE APC

FIG. 8A

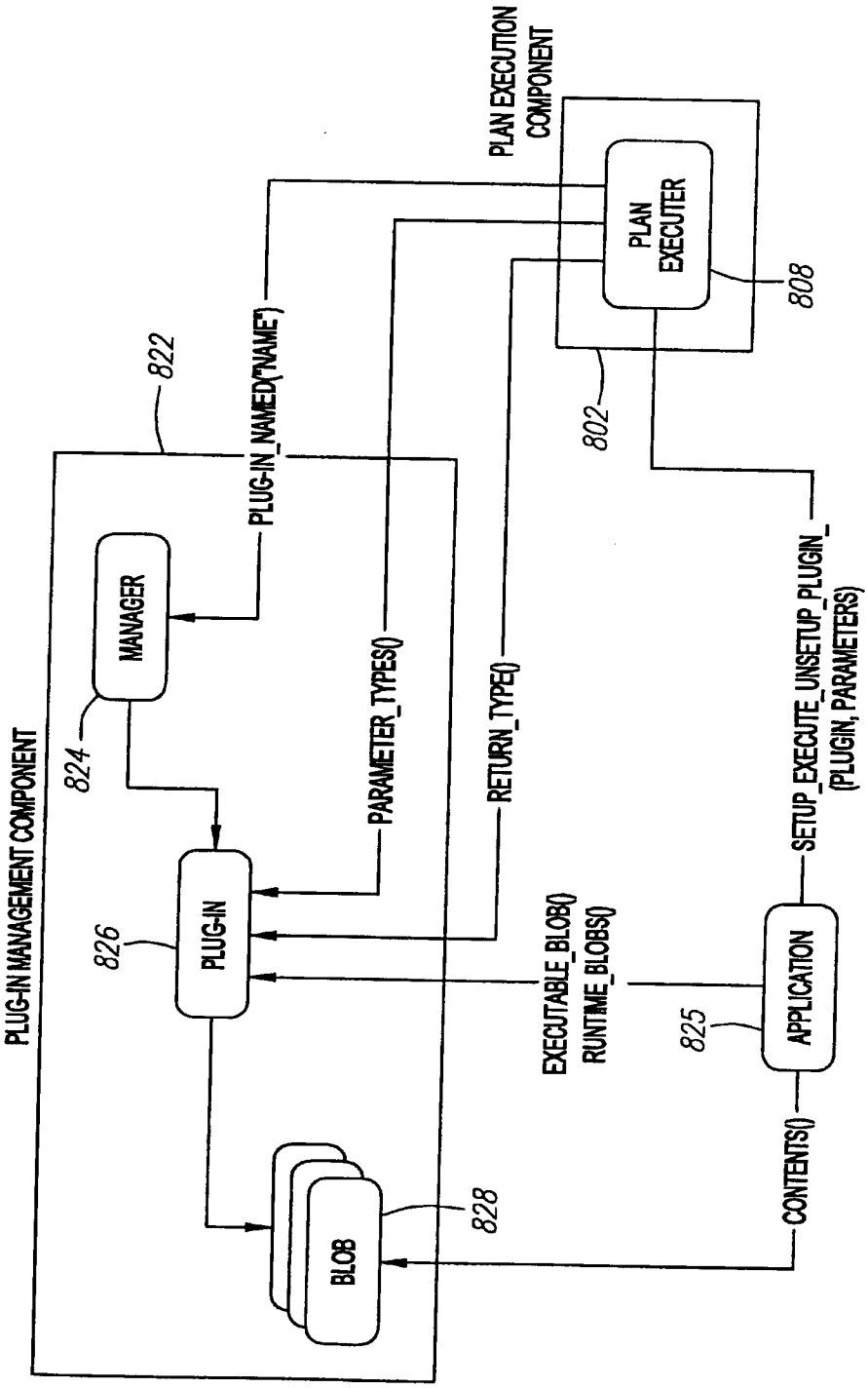

PLAN EXECUTION COMPONENT
PLAN EXECUTER
808
802
822
PLUG-IN_NAMED("NAME")
MANAGER
824
PARAMETER_TYPES()
RETURN_TYPE()
SETUP_EXECUTE_UNSETUP_PLUGIN_
(PLUGIN, PARAMETERS)
PLUG-IN MANAGEMENT COMPONENT
826
PLUG-IN
EXECUTABLE_BLOB()
RUNTIME_BLOBS()
825
APPLICATION
828
BLOB
CONTENTS()

FIG. 8B

FIG. 8C

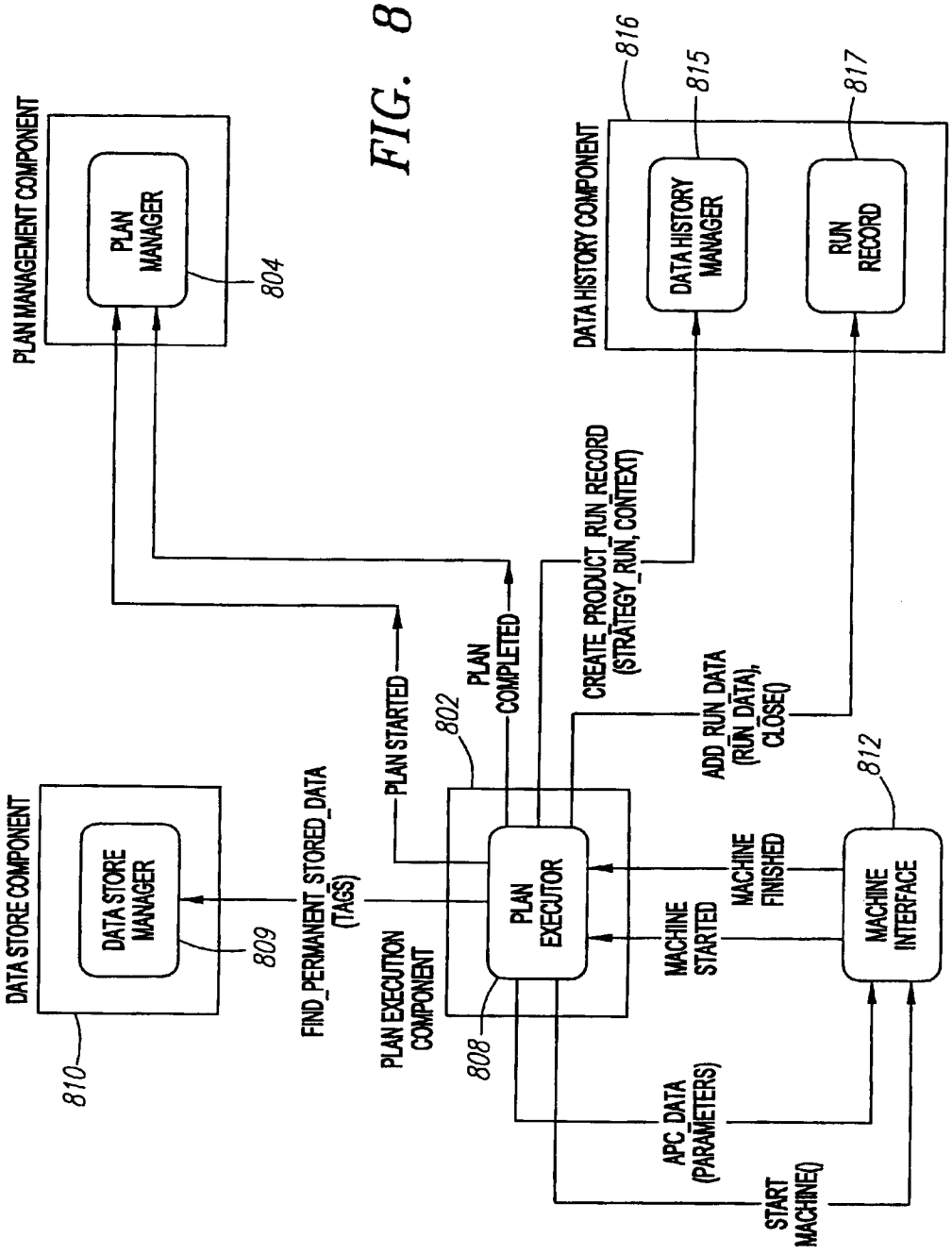
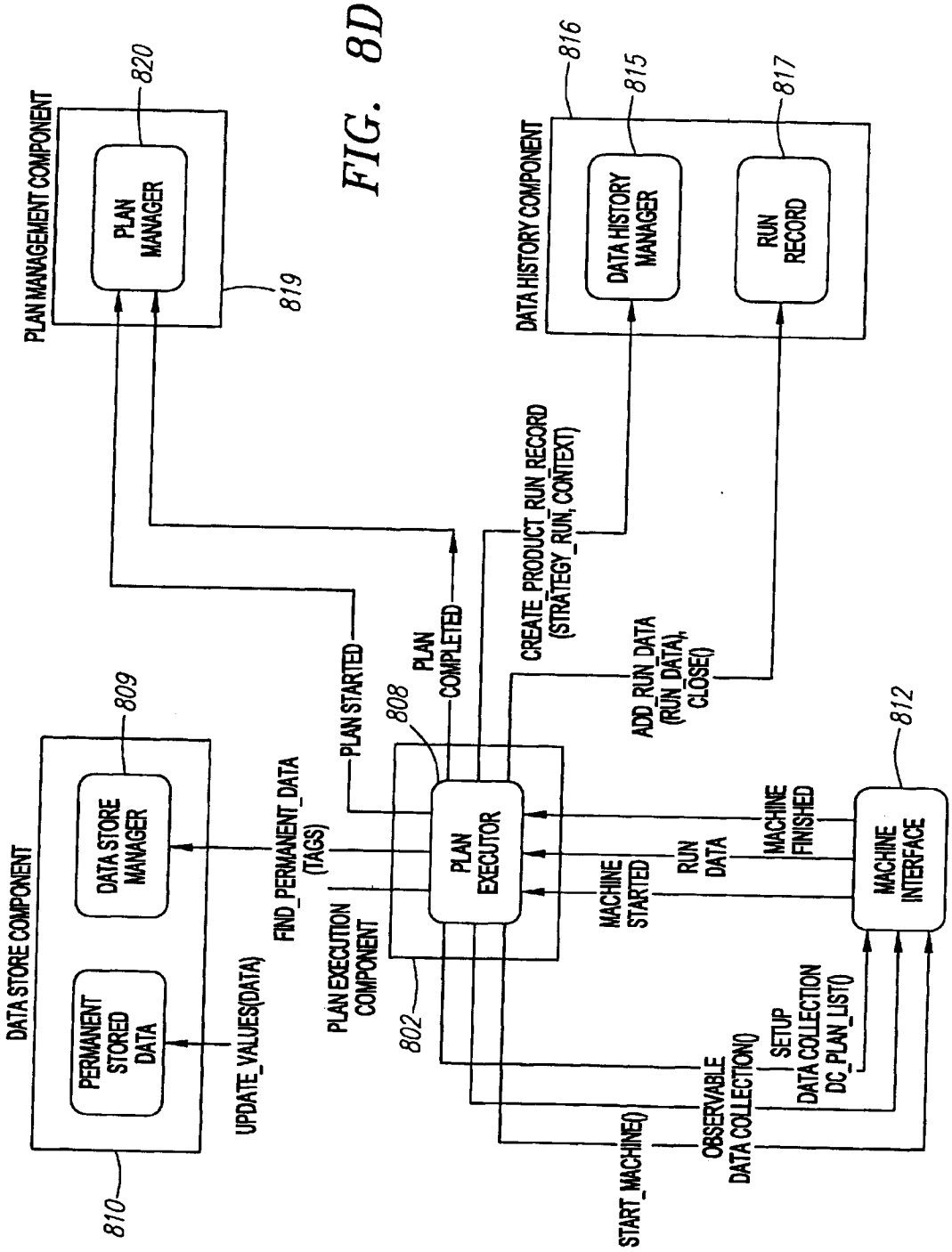

PLAN MANAGEMENT COMPONENT
PLAN MANAGER
804
DATA HISTORY COMPONENT
DATA HISTORY MANAGER
RUN RECORD
816
815
817
DATA STORE COMPONENT
DATA STORE MANAGER
809
810
PLAN EXECUTION COMPONENT
PLAN EXECUTOR
808
802
812
MACHINE INTERFACE
PLAN STARTED
PLAN COMPLETED
CREATE_PRODUCT_RUN_RECORD (STRATEGY_RUN, CONTEXT)
ADD RUN DATA (RUN DATA), CLOSE()
FIND_PERMANENT_STORED_DATA (TAGS)
MACHINE STARTED
MACHINE FINISHED
APC DATA (PARAMETERS)
START MACHINE()

FIG. 8D


PLAN MANAGEMENT COMPONENT
820
PLAN MANAGER
819
DATA HISTORY COMPONENT
816
815
817
DATA HISTORY MANAGER
RUN RECORD
CREATE_PRODUCT_RUN_RECORD (STRATEGY_RUN_CONTEXT)
ADD_RUN_DATA (RUN_DATA), CLOSE()
PLAN STARTED
PLAN COMPLETED
808
809
DATA STORE COMPONENT
DATA STORE MANAGER
PERMANENT STORED DATA
FIND_PERMANENT_DATA (TAGS)
UPDATE_VALUES(DATA)
PLAN EXECUTOR
PLAN EXECUTION COMPONENT
802
MACHINE STARTED
RUN DATA
MACHINE FINISHED
812
MACHINE INTERFACE
SETUP DATA COLLECTION DC_PLAN_LIST()
OBSERVABLE DATA COLLECTION()
START_MACHINE()
810

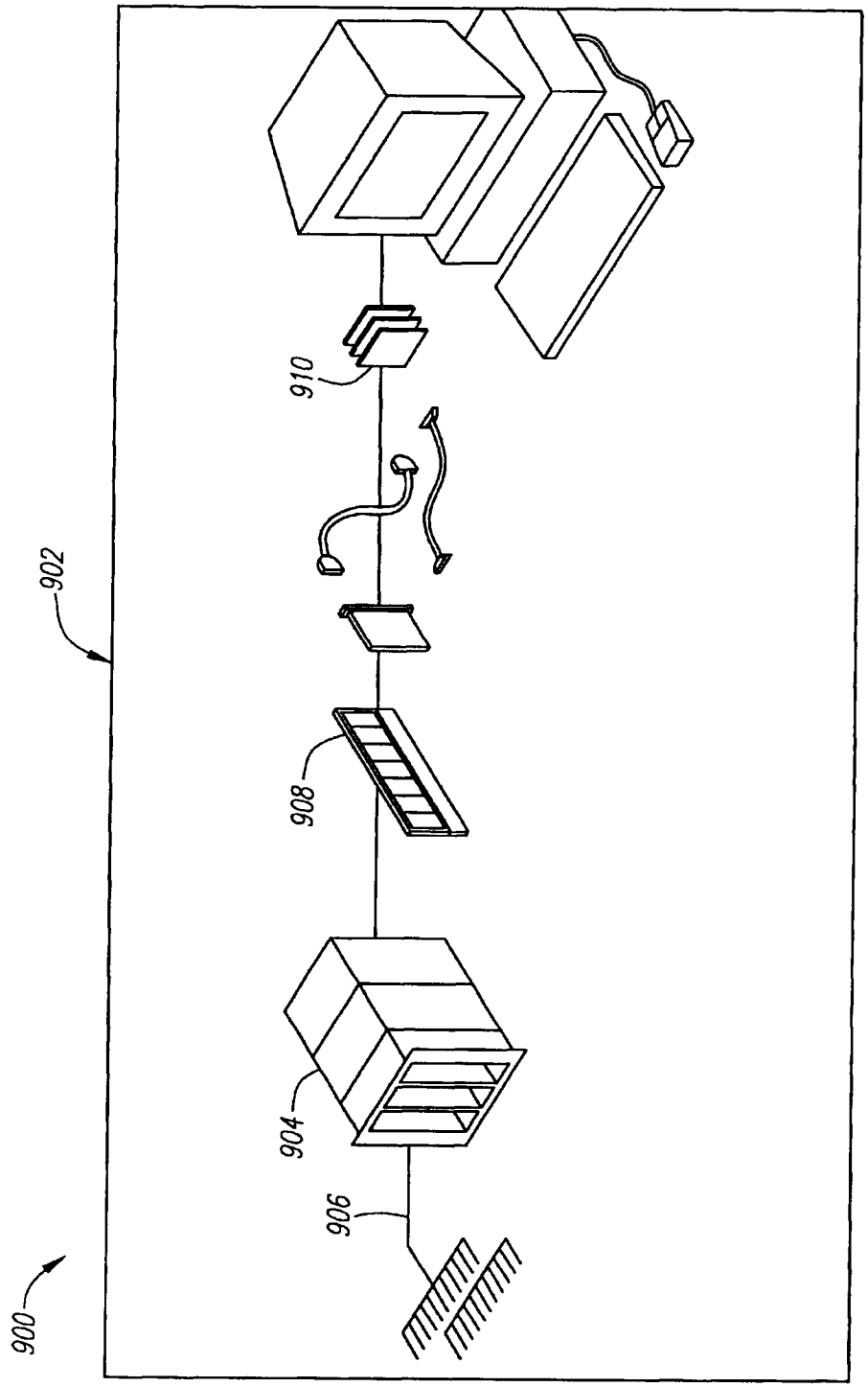



FIG. 9

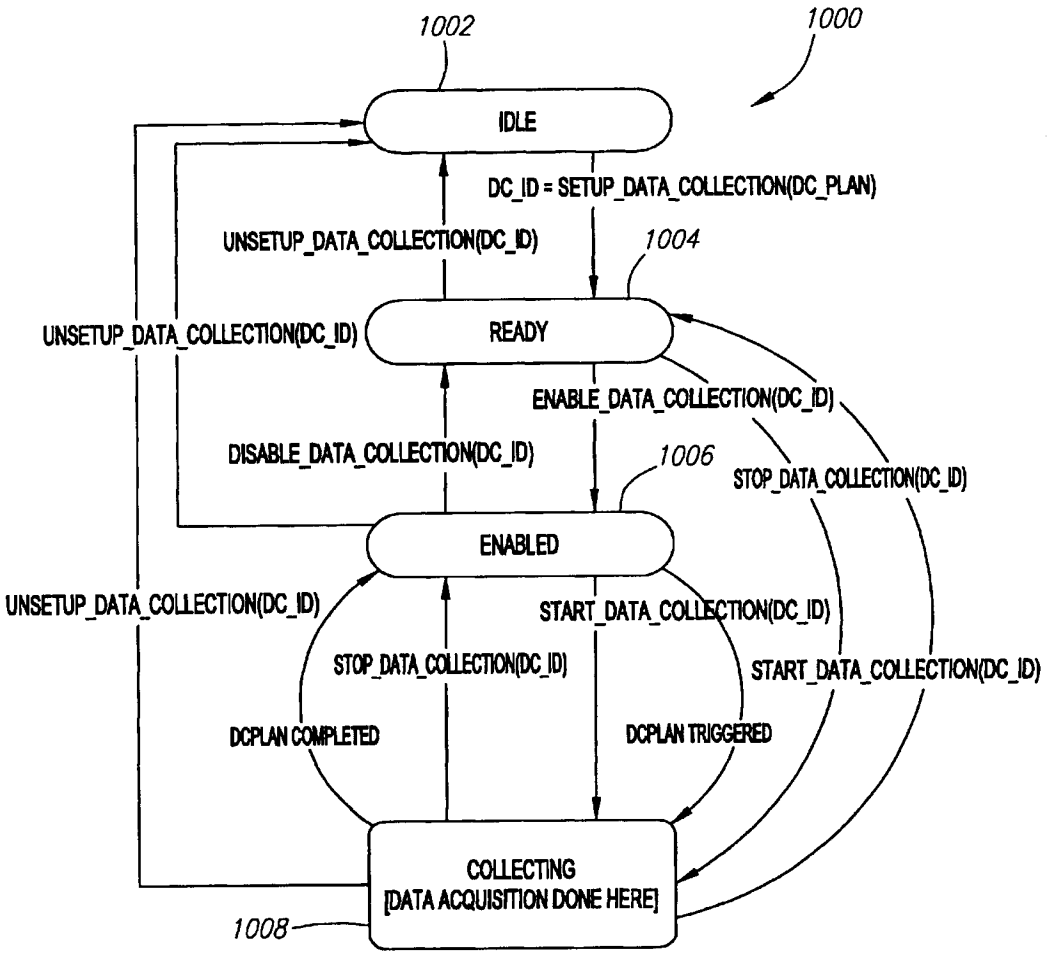

1002
1000
IDLE
DC_ID = SETUP_DATA_COLLECTION(DC_PLAN)
UNSETUP_DATA_COLLECTION(DC_ID)
1004
UNSETUP_DATA_COLLECTION(DC_ID)
READY
ENABLE_DATA_COLLECTION(DC_ID)
DISABLE_DATA_COLLECTION(DC_ID)
1006
STOP_DATA_COLLECTION(DC_ID)
ENABLED
UNSETUP_DATA_COLLECTION(DC_ID)
START_DATA_COLLECTION(DC_ID)
STOP_DATA_COLLECTION(DC_ID)
START_DATA_COLLECTION(DC_ID)
DCPLAN COMPLETED
DCPLAN TRIGGERED
COLLECTING
[DATA ACQUISITION DONE HERE]
1008

*FIG. 10*

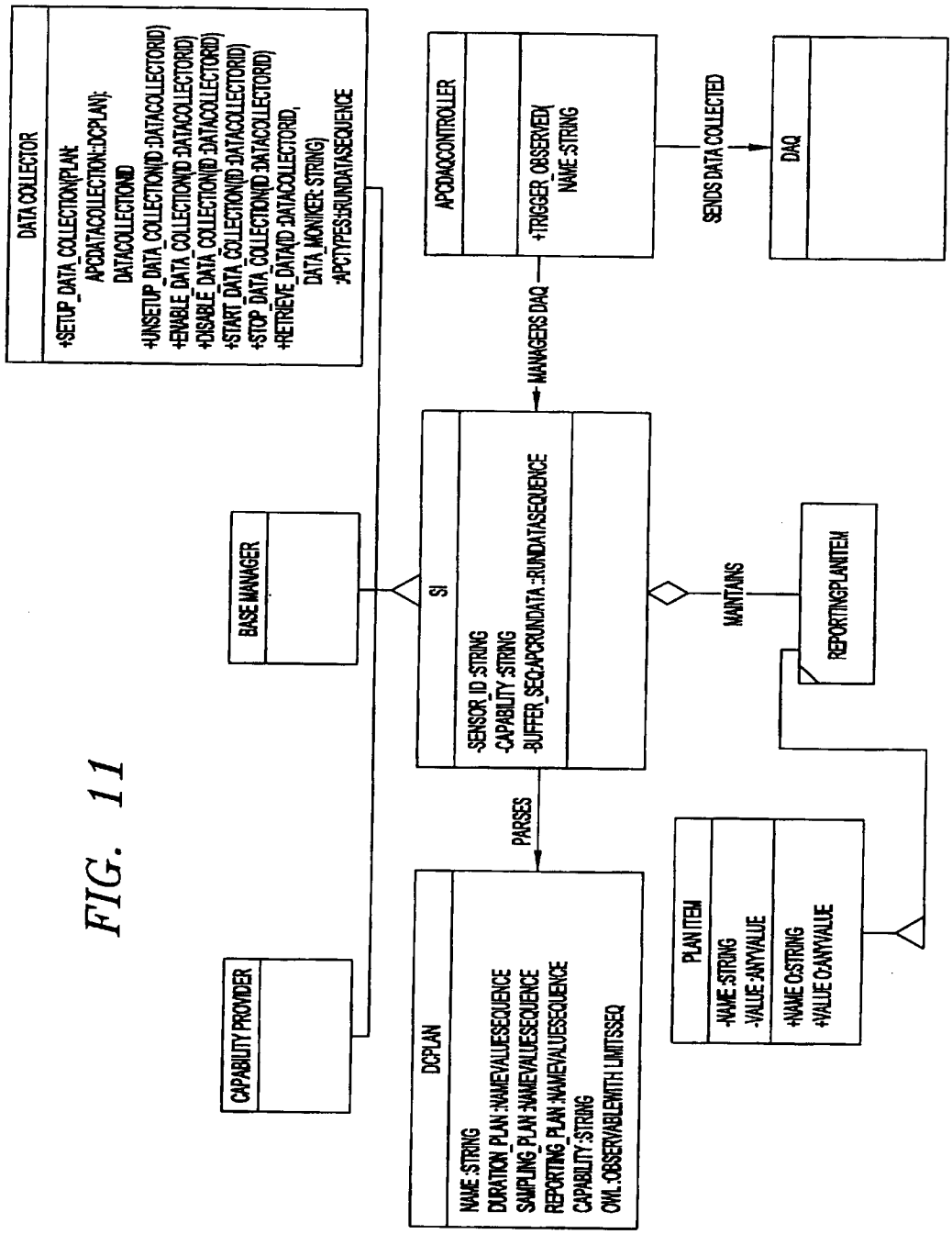

DATA COLLECTOR
+SETUP_DATA_COLLECTION(PLAN: APCDATACOLLECTION::DCPLAN): DATACOLLECTIONID
+UNSETUP_DATA_COLLECTION(ID :DATACOLLECTORID)
+ENABLE_DATA_COLLECTION(ID :DATACOLLECTORID)
+DISABLE_DATA_COLLECTION(ID :DATACOLLECTORID)
+START_DATA_COLLECTION(ID :DATACOLLECTORID)
+STOP_DATA_COLLECTION(ID :DATACOLLECTORID)
+RETRIEVE_DATA(ID :DATACOLLECTORID, DATA_MONIKER: STRING) :APCTYPES::RUNDATASEQUENCE
APCDAQCONTROLLER
+TRIGGER_OBSERVED( NAME :STRING
SENDS DATA COLLECTED
DAQ
MANAGERS DAQ
BASE MANAGER
SI
-SENSOR_ID :STRING
-CAPABILITY :STRING
-BUFFER_SEQ:APCRUNDATA ::RUNDATASEQUENCE
MAINTAINS
REPORTINGPLANITEM
PARSES
CAPABILITY PROVIDER
DCPLAN
NAME :STRING
DURATION PLAN :NAMEVALUESEQUENCE
SAMPLING_PLAN :NAMEVALUESEQUENCE
REPORTING_PLAN :NAMEVALUESSEQUENCE
CAPABILITY :STRING
OWL::OBSERVABLEWITH LIMITSSEQ
PLANITEM
+NAME :STRING
-VALUE :ANYVALUE
+NAME OSTRING
+VALUE OANYVALUE

*FIG. 11*

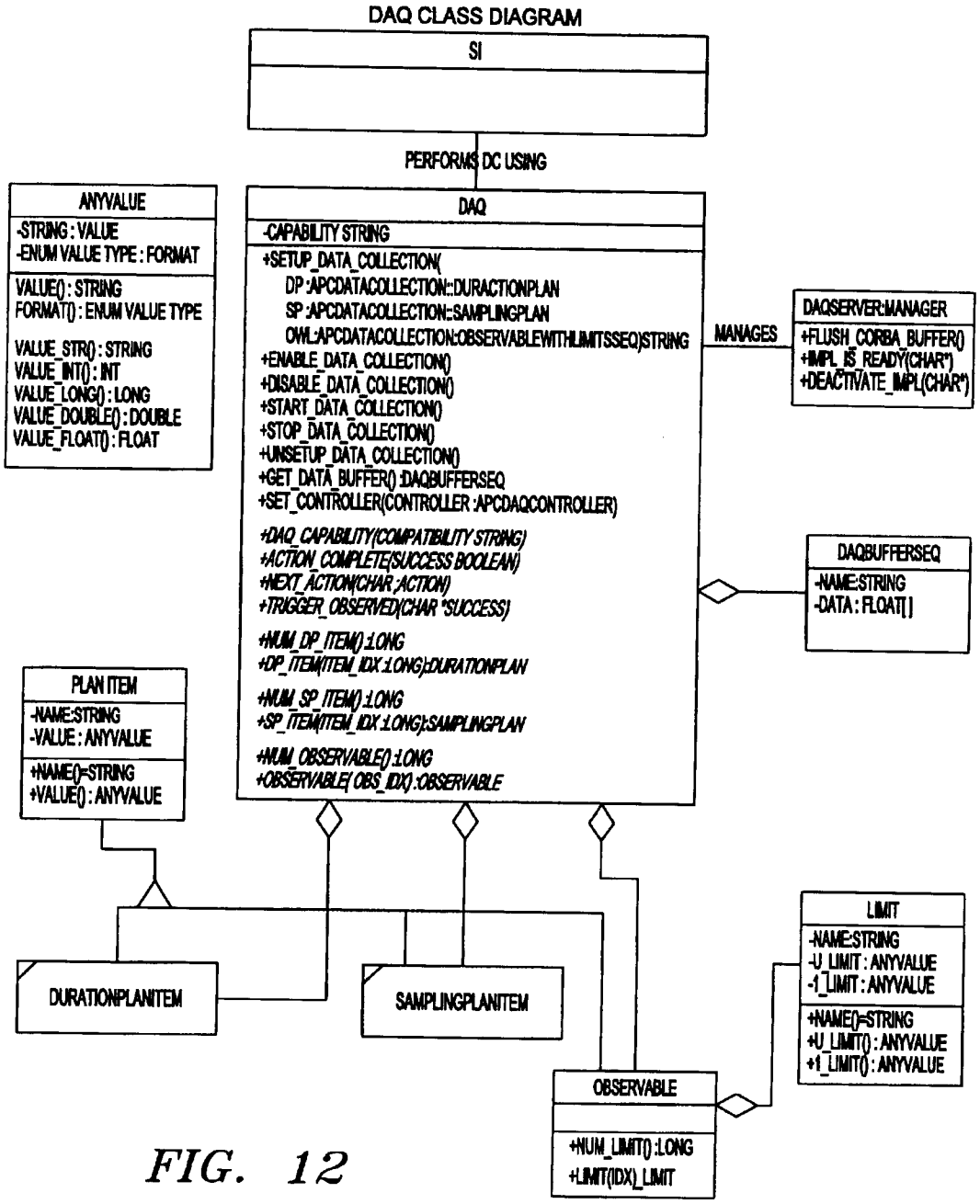

DAQ CLASS DIAGRAM
SI
PERFORMS DC USING
ANYVALUE
-STRING : VALUE
-ENUM VALUE TYPE : FORMAT
VALUE() : STRING
FORMAT() : ENUM VALUE TYPE
VALUE_STR() : STRING
VALUE_INT() : INT
VALUE_LONG() : LONG
VALUE_DOUBLE() : DOUBLE
VALUE_FLOAT() : FLOAT
DAQ
-CAPABILITY STRING
+SETUP_DATA_COLLECTION(
DP :APCDATACOLLECTION::DURACTIONPLAN
SP :APCDATACOLLECTION::SAMPLINGPLAN
OWL:APCDATACOLLECTION:OBSERVABLEWITHLIMITSSEQ)STRING
+ENABLE_DATA_COLLECTION()
+DISABLE_DATA_COLLECTION()
+START_DATA_COLLECTION()
+STOP_DATA_COLLECTION()
+UNSETUP_DATA_COLLECTION()
+GET_DATA_BUFFER() :DAQBUFFERSEQ
+SET_CONTROLLER(CONTROLLER :APCDAQCONTROLLER)
+DAQ_CAPABILITY(COMPATIBILITY STRING)
+ACTION_COMPLETE(SUCCESS BOOLEAN)
+NEXT_ACTION(CHAR :ACTION)
+TRIGGER_OBSERVED(CHAR *SUCCESS)
+NUM_DP_ITEM() :LONG
+DP_ITEM(ITEM_IDX :LONG):DURATIONPLAN
+NUM_SP_ITEM() :LONG
+SP_ITEM(ITEM_IDX :LONG):SAMPLINGPLAN
+NUM_OBSERVABLE() :LONG
+OBSERVABLE( OBS_IDX) :OBSERVABLE
MANAGES
DAQSERVER:MANAGER
+FLUSH_CORBA_BUFFER()
+IMPL_IS_READY(CHAR*)
+DEACTIVATE_IMPL(CHAR*)
DAQBUFFERSEQ
-NAME:STRING
-DATA : FLOAT[ ]
PLAN ITEM
-NAME:STRING
-VALUE : ANYVALUE
+NAME()=STRING
+VALUE() : ANYVALUE
DURATIONPLANITEM
SAMPLINGPLANITEM
LIMIT
-NAME:STRING
-U_LIMIT : ANYVALUE
-1_LIMIT : ANYVALUE
+NAME()=STRING
+U_LIMIT() : ANYVALUE
+1_LIMIT() : ANYVALUE
OBSERVABLE
+NUM_LIMIT() :LONG
+LIMIT(IDX)_LIMIT

*FIG. 12*

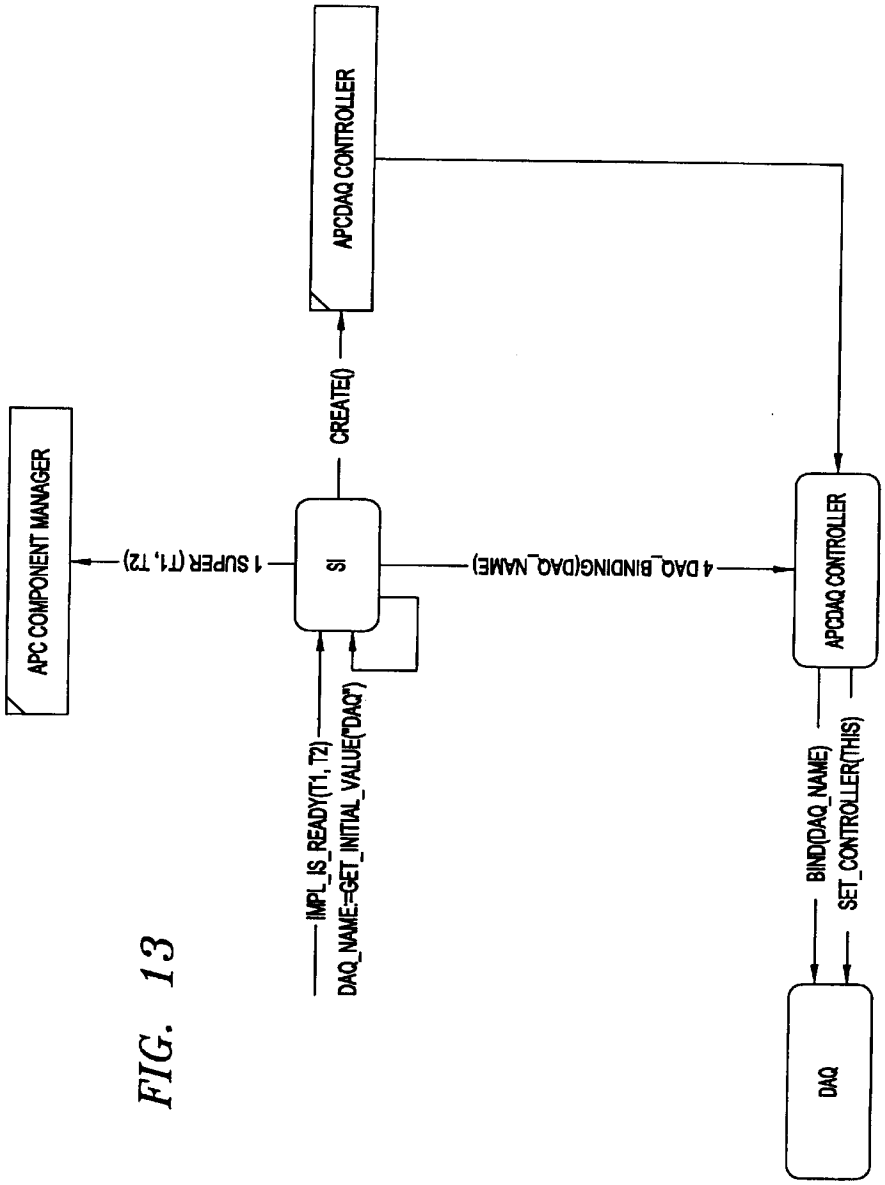

APCDAQ CONTROLLER
CREATE()
APC COMPONENT MANAGER
1 SUPER (T1, T2)
SI
4 DAQ_BINDING(DAQ_NAME)
APCDAQ CONTROLLER
IMPL_IS_READY(T1, T2)
DAQ_NAME=GET_INITIAL_VALUE("DAQ")
BIND(DAQ_NAME)
SET_CONTROLLER(THIS)
DAQ

FIG. 13

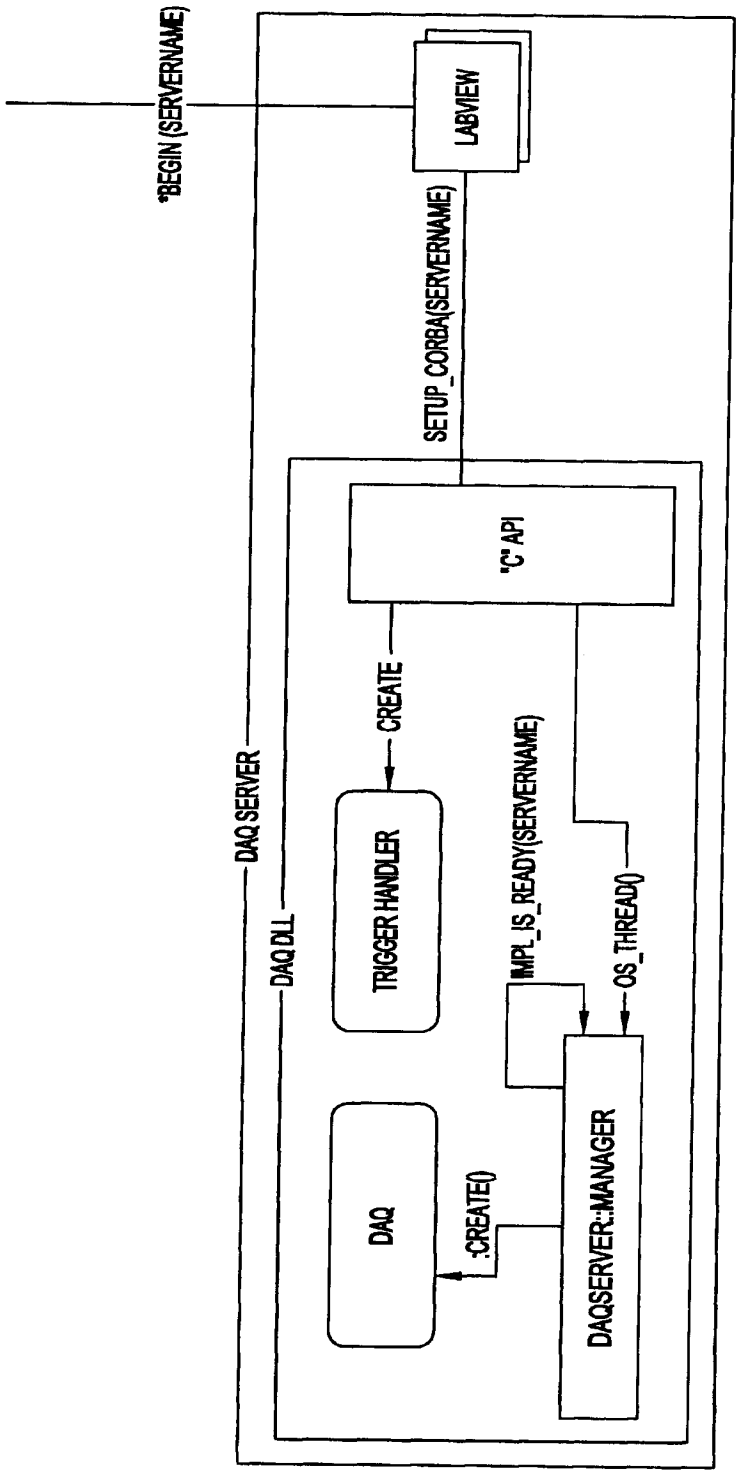

"BEGIN (SERVERNAME)
LABVIEW
SETUP_CORBA(SERVERNAME)
"C" API
CREATE
DAQ SERVER
DAQ DLL
TRIGGER HANDLER
IMPL_IS_READY(SERVERNAME)
OS_THREAD()
DAQ
:CREATE()
DAQSERVER::MANAGER

FIG. 14

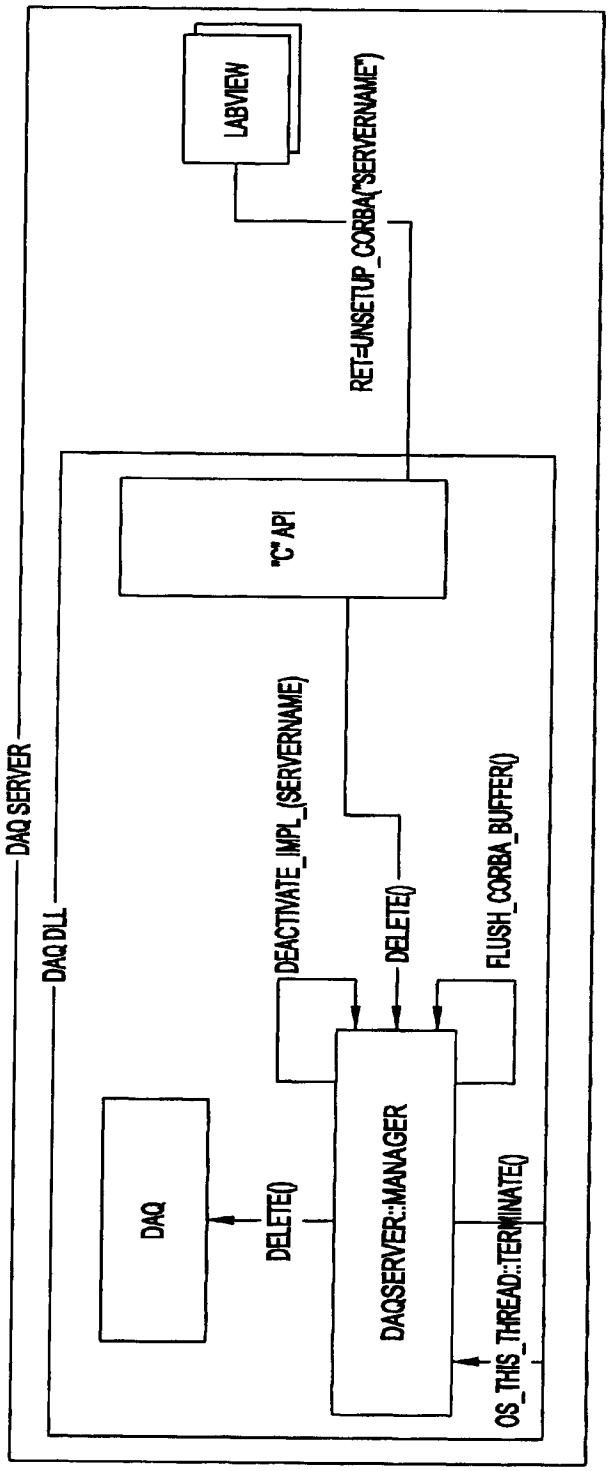

LABVIEW
RET=UNSETUP_CORBA("SERVERNAME")
"C" API
DAQ SERVER
DAQ DLL
DEACTIVATE_IMPL_(SERVERNAME)
DELETE()
FLUSH_CORBA_BUFFER()
DAQ
DELETE()
DAQSERVER::MANAGER
OS_THIS_THREAD::TERMINATE()

FIG. 15

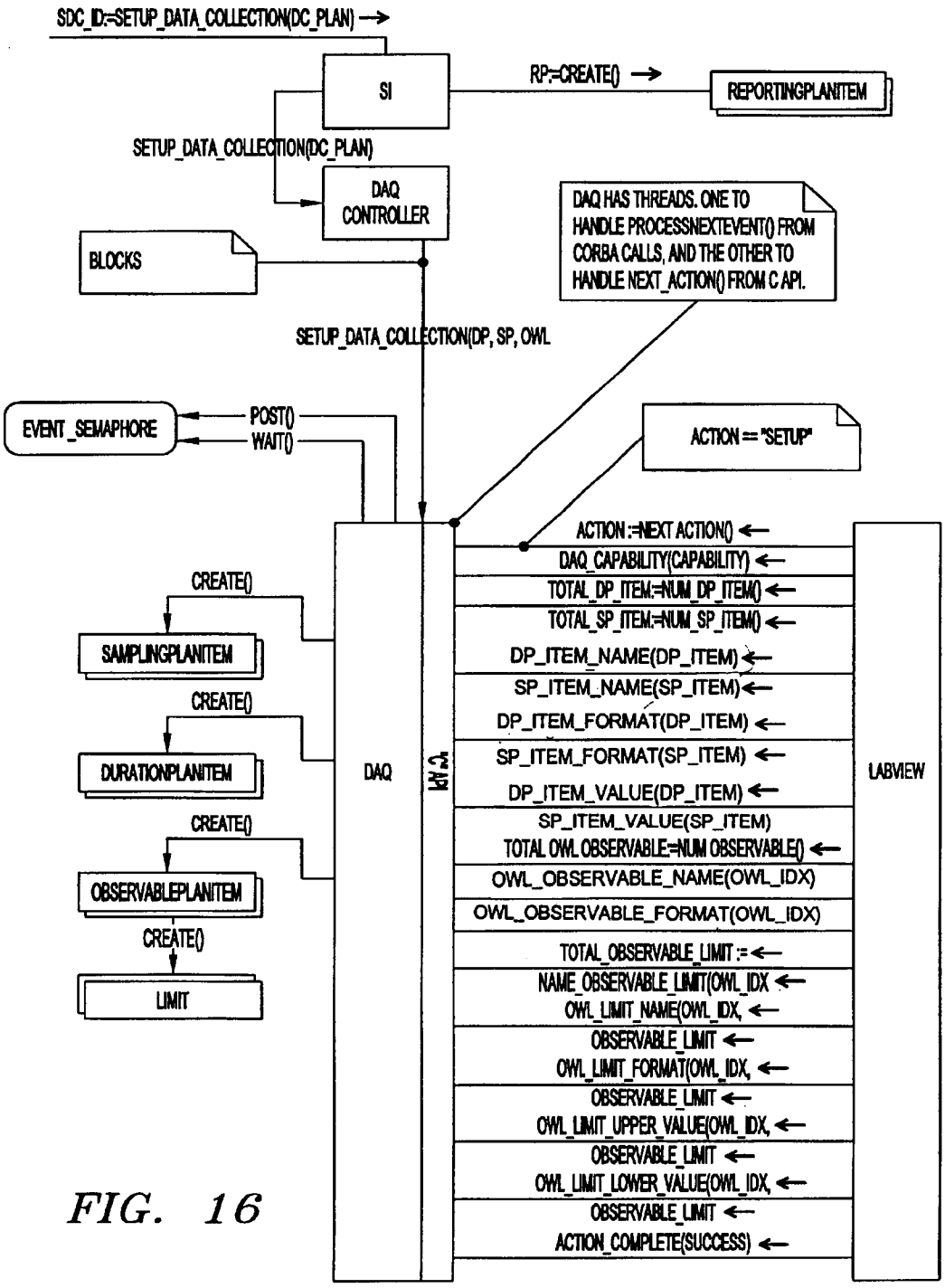

SDC_ID:=SETUP_DATA_COLLECTION(DC_PLAN)
SI
RP:=CREATE()
REPORTINGPLANITEM
SETUP_DATA_COLLECTION(DC_PLAN)
DAQ CONTROLLER
DAQ HAS THREADS. ONE TO HANDLE PROCESSNEXTEVENT() FROM CORBA CALLS, AND THE OTHER TO HANDLE NEXT_ACTION() FROM C API.
BLOCKS
SETUP_DATA_COLLECTION(DP, SP, OWL
EVENT_SEMAPHORE
POST()
WAIT()
ACTION == "SETUP"
ACTION :=NEXT ACTION()
DAQ_CAPABILITY(CAPABILITY)
TOTAL_DP_ITEM:=NUM_DP_ITEM()
TOTAL_SP_ITEM:=NUM_SP_ITEM()
DP_ITEM_NAME(DP_ITEM)
SP_ITEM_NAME(SP_ITEM)
DP_ITEM_FORMAT(DP_ITEM)
SP_ITEM_FORMAT(SP_ITEM)
DP_ITEM_VALUE(DP_ITEM)
SP_ITEM_VALUE(SP_ITEM)
TOTAL OWL OBSERVABLE:=NUM OBSERVABLE()
OWL_OBSERVABLE_NAME(OWL_IDX)
OWL_OBSERVABLE_FORMAT(OWL_IDX)
TOTAL_OBSERVABLE_LIMIT :=
NAME_OBSERVABLE_LIMIT(OWL_IDX
OWL_LIMIT_NAME(OWL_IDX,
OBSERVABLE_LIMIT
OWL_LIMIT_FORMAT(OWL_IDX,
OBSERVABLE_LIMIT
OWL_LIMIT_UPPER_VALUE(OWL_IDX,
OBSERVABLE_LIMIT
OWL_LIMIT_LOWER_VALUE(OWL_IDX,
OBSERVABLE_LIMIT
ACTION_COMPLETE(SUCCESS)
CREATE()
SAMPLINGPLANITEM
CREATE()
DURATIONPLANITEM
CREATE()
OBSERVABLEPLANITEM
CREATE()
LIMIT
DAQ
IDL
LABVIEW

*FIG. 16*

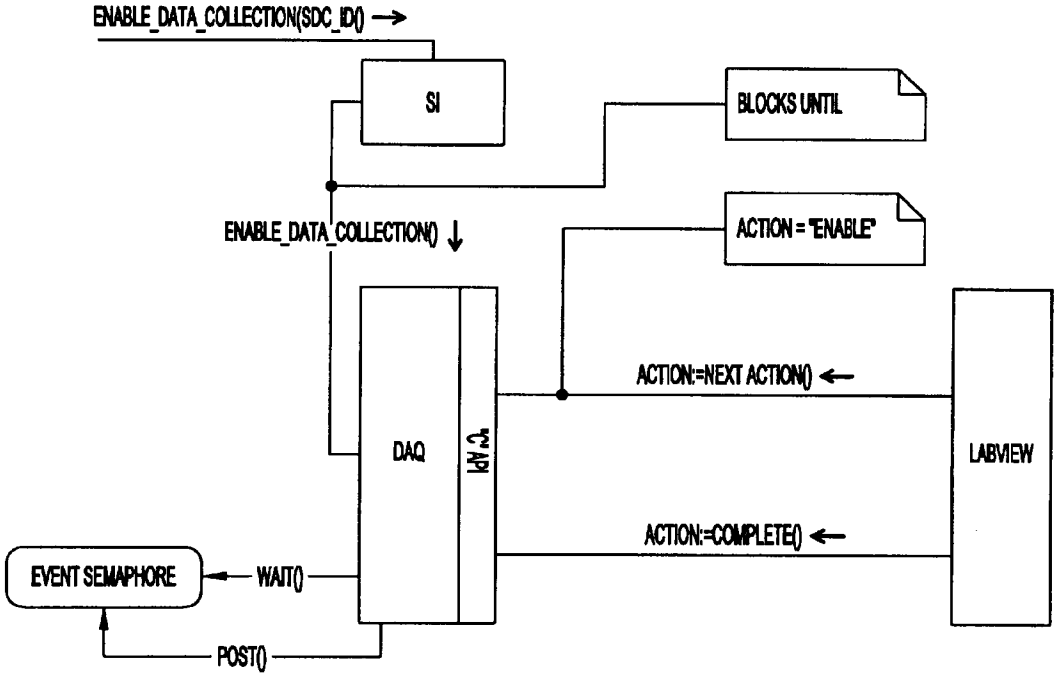

ENABLE_DATA_COLLECTION(SDC_ID()
SI
BLOCKS UNTIL
ENABLE_DATA_COLLECTION()
ACTION = "ENABLE"
DAQ
C API
ACTION:=NEXT ACTION()
LABVIEW
ACTION:=COMPLETE()
EVENT SEMAPHORE
WAIT()
POST()

FIG. 17

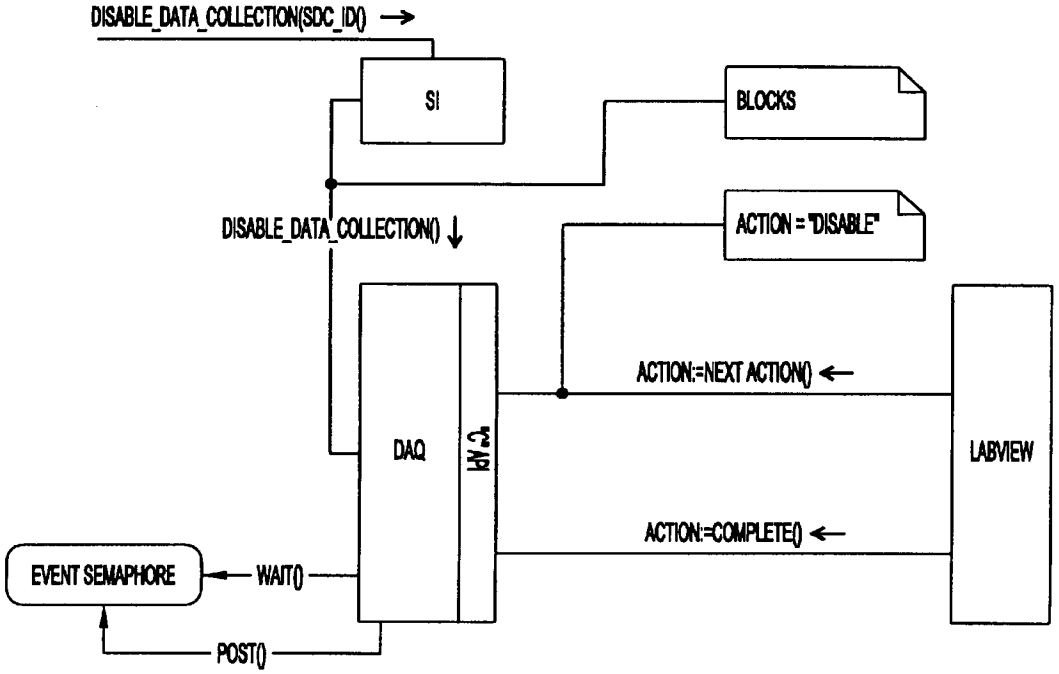

DISABLE_DATA_COLLECTION(SDC_ID()
SI
BLOCKS
DISABLE_DATA_COLLECTION()
ACTION = "DISABLE"
DAQ
C API
ACTION:=NEXT ACTION()
LABVIEW
ACTION:=COMPLETE()
EVENT SEMAPHORE
WAIT()
POST()

FIG. 18

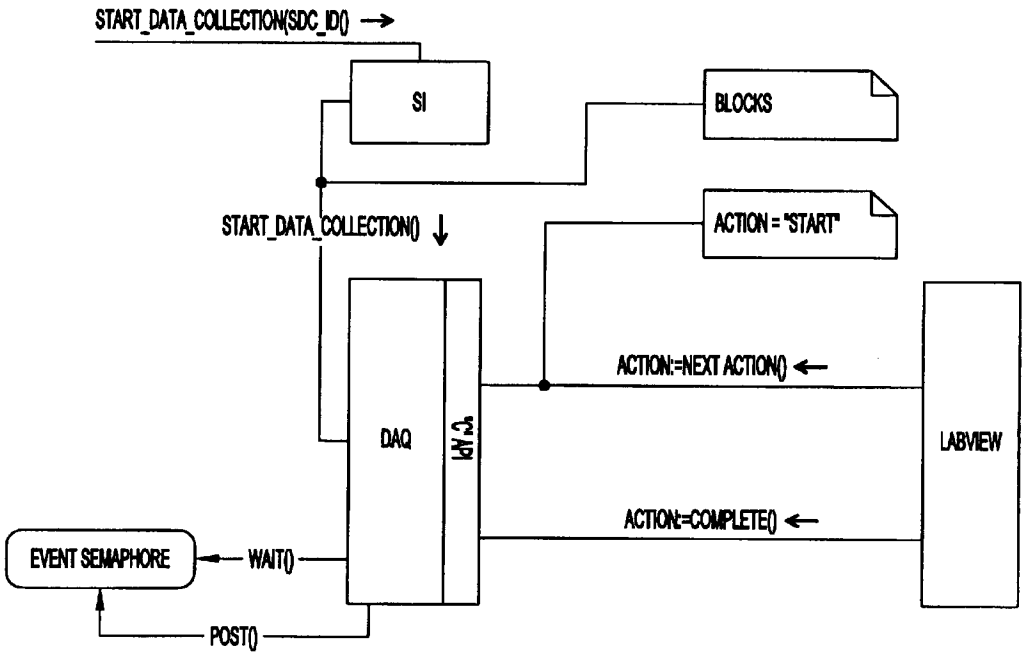

START_DATA_COLLECTION(SDC_ID() →
SI
BLOCKS
START_DATA_COLLECTION() ↓
ACTION = "START"
DAQ
LabVIEW C API
ACTION:=NEXT ACTION() ←
LABVIEW
ACTION:=COMPLETE() ←
EVENT SEMAPHORE
WAIT()
POST()

FIG. 19

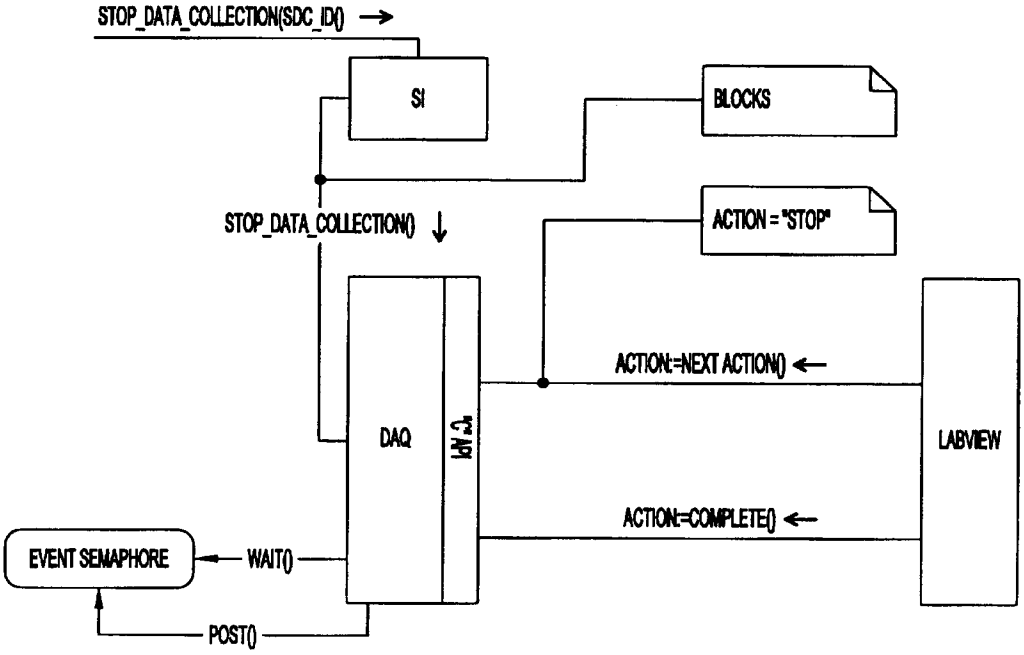

STOP_DATA_COLLECTION(SDC_ID() →
SI
BLOCKS
STOP_DATA_COLLECTION() ↓
ACTION = "STOP"
DAQ
LabVIEW C API
ACTION:=NEXT ACTION() ←
LABVIEW
ACTION:=COMPLETE() ←
EVENT SEMAPHORE
WAIT()
POST()

FIG. 20

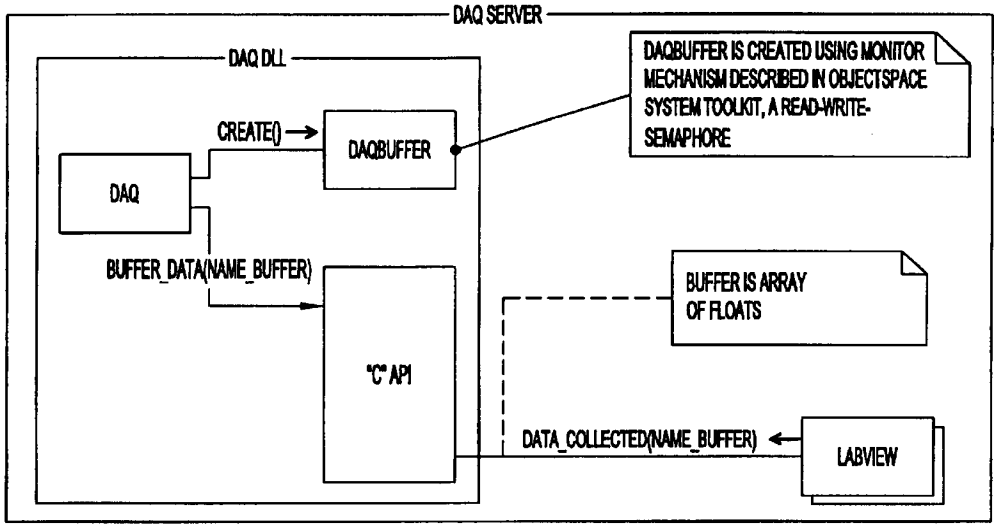

DAQ SERVER
DAQ DLL
DAQBUFFER IS CREATED USING MONITOR MECHANISM DESCRIBED IN OBJECTSPACE SYSTEM TOOLKIT, A READ-WRITE-SEMAPHORE
CREATE()
DAQBUFFER
DAQ
BUFFER_DATA(NAME_BUFFER)
"C" API
BUFFER IS ARRAY OF FLOATS
DATA_COLLECTED(NAME_BUFFER)
LABVIEW

*FIG. 21*

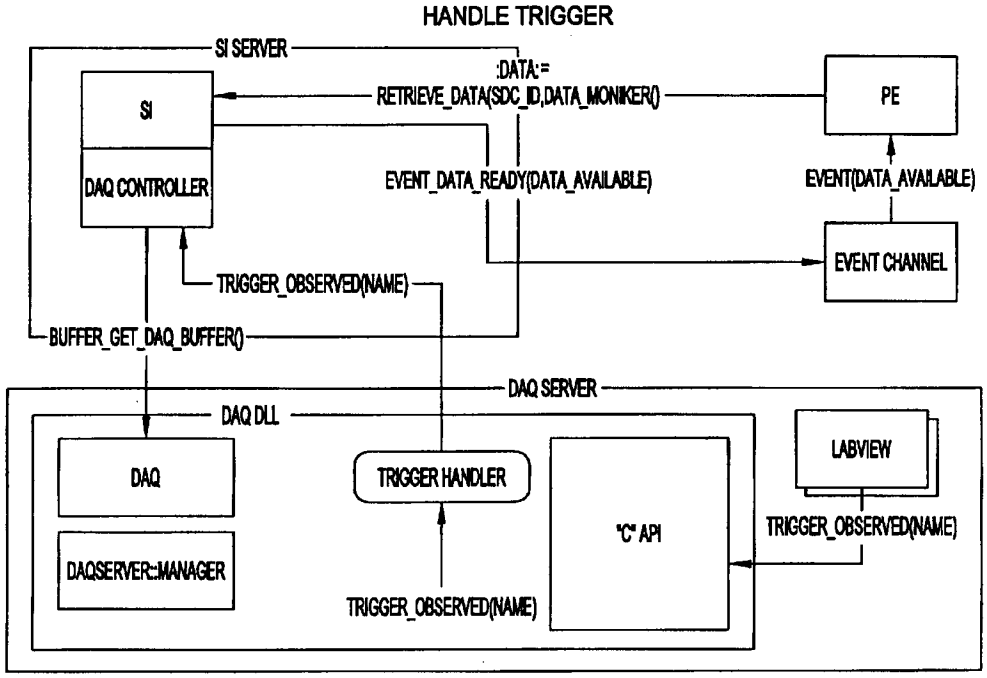

HANDLE TRIGGER
SI SERVER
:DATA:=
SI
RETRIEVE_DATA(SDC_ID,DATA_MONIKER()
PE
DAQ CONTROLLER
EVENT_DATA_READY(DATA_AVAILABLE)
EVENT(DATA_AVAILABLE)
EVENT CHANNEL
TRIGGER_OBSERVED(NAME)
BUFFER_GET_DAQ_BUFFER()
DAQ SERVER
DAQ DLL
DAQ
TRIGGER HANDLER
"C" API
LABVIEW
TRIGGER_OBSERVED(NAME)
DAQSERVER::MANAGER
TRIGGER_OBSERVED(NAME)

*FIG. 22*

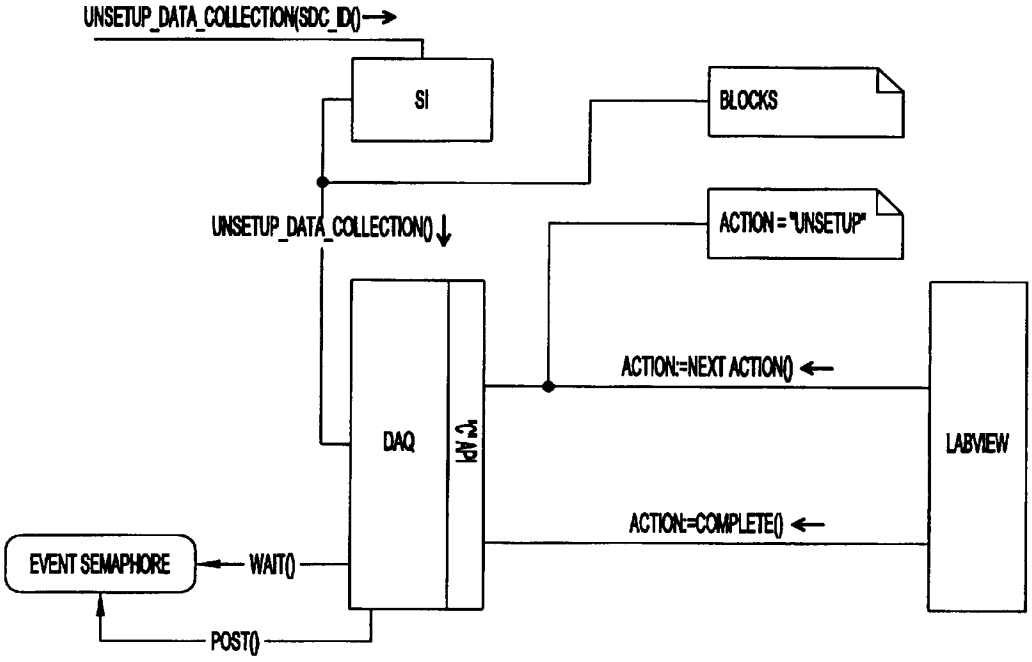

UNSETUP_DATA_COLLECTION(SDC_ID()
SI
BLOCKS
UNSETUP_DATA_COLLECTION()
ACTION = "UNSETUP"
ACTION:=NEXT ACTION()
DAQ
LabVIEW
ACTION:=COMPLETE()
EVENT SEMAPHORE
WAIT()
POST()

FIG. 23

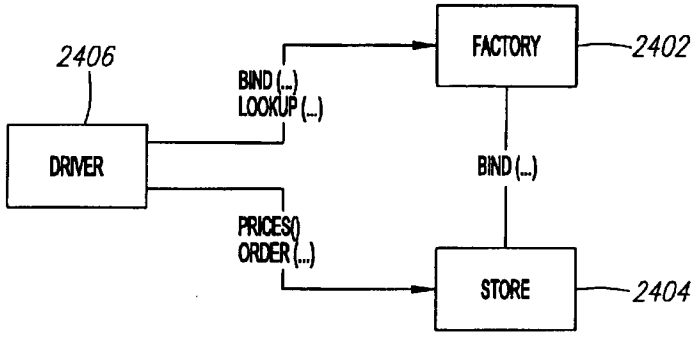

2406
FACTORY
2402
BIND (...)
LOOKUP (...)
DRIVER
BIND (...)
PRICES()
ORDER (...)
STORE
2404

FIG. 24

# ADVANCED PROCESS CONTROL FOR SEMICONDUCTOR MANUFACTURING

## BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to process control systems including computer-based materials management. More precisely, the present invention relates to a feed forward process control system used in semiconductor fabrication based on material groupings.

2. Description of the Related Art

In the past, semiconductor manufacturing process control was largely achieved by ensuring that process parameters were set on a machine controller according to machine-dependent recipes. The basic philosophy of conventional semiconductor manufacturing process control is that if all settings that affect the process are set correctly, the machine will consistently produce a specified product. Using the conventional approach, manufacturing personnel act to relate machine settings to product characteristics. The approach has yet to be fully realized, however, due to a number of factors, including variability in equipment performance, variability in incoming materials such as wafers and chemicals, increasingly complex processes, and a lack of adequate models relating process settings to product characteristics. Success using the conventional process control approach becomes much less likely as the size of wafer features becomes smaller.

Engineers have derived recipe settings based largely on experience, intuition, and, more recently, Response Surface Methodology (RSM) experiments. Initially, the recipes were manually downloaded to the equipment by operators/technicians. Subsequently, Factory Control Systems incorporating Equipment Integration (EI) functionality provided automated recipe management and download operations.

Most recently, engineers have used Statistical Process Controls (SPC) concepts and methods for monitoring the performance of processes to verify that a process remains in a state of "statistical control." Initially, operators and technicians performed SPC manually. Subsequently, all-manual charting was replaced with computerized factory control systems (FCS)-based SPC charts. In some cases automated Trouble Shooting Guides (TSGs) supplied automation to process control tasks. SPC is a fault detection methodology. TSGs perform rule-based classification and assist with problem resolution. SPC helps distinguish between two types of process variation: common and special. SPC out-of-control signals are clues that are useful for identifying sources of special variation. Once a cause for special variation is determined, manufacturing can produce improvements in the process and product quality. As a fault detection and classification methodology, SPC relies upon an intimate understanding of the process and is largely manual and reactive. FIG. 1 is a schematic block diagram that shows a traditional SPC process.

The conventional process control approaches have resulted in substantial progress. However, reactive process control techniques such as SPC do not achieve and sustain desired product yields and resource productivity, particularly in light of the size and speed specifications of future products. The semiconductor industry must continue to develop and deploy new process control methods. To address significant unresolved problems, an Advanced Process Control (APC) Framework is needed with the ability to provide:

(1) Sensor-based automated fault detection to provide in real time the equipment or process conditions that result in a misprocessed wafer.



(2) Classification of detected faults to determine the cause of a fault and expedite repair of a tool.

(3) Model-based run-to-run process control using sensor inputs, process models, and process control strategies to ensure that the process remains optimal for every die on every wafer.

(4) Model-based real-time process control using in situ inputs, process models, and process control strategies to correctly process control parameters during the process run, ensuring that product characteristics are achieved.

Common Object Request Broker Architecture (CORBA) based technologies are used as a communication interface between clients and servers, often in highly complex systems. Due to the complex nature of the systems, testing can be difficult. The system complexity arises because multiple components interact with one another over a network, which introduces problems. Many components operate both as a client and as a server since, in servicing a request, a component calls other remote services which, in turn, call other remote services. Testing of a client-server component is difficult since the component uses a driver to send requests and to send harnesses to emulate other components that interact with the client-server component. Also failures and performance problems may occur in any of multiple potentially remote components and therefore be difficult to isolate.

The problems and complexities of these technologies including complexities arising from the integration of multiple components, the verification of the correct operation of all of the multiple components individually and while interacting, and the analysis not only of correct operation but also of performance place a high demand on system test personnel. No longer can the least experienced developers perform the testing. More experienced architects are needed to specify and set up a testing infrastructure and perform the tests.

What is needed is a strategy and technique that improves the management and prospects for success of the testing process.

Present-day semiconductor manufacturing environments include the following characteristics that limit the ability of an environment to support the manufacture of complex, high-value products. First, stand-alone equipment controllers have limited communications capability and limited provisions for external process control. Second, semiconductor manufacturing environments are limited by "static" (nonadaptive) process control approaches. Furthermore, present-day semiconductor manufacturing environments lack models to support the development and use of control algorithms. In addition the manufacturing environments are supplied by nonuniform, disparate, and incomplete sources of manufacturing data for driving process control algorithms. Closed, monolithic factory system architectures prevent integration of new capabilities, especially from multiple suppliers.

## SUMMARY OF THE INVENTION

In accordance with an aspect of the present invention, an Advanced Process Control (APC) framework, which is based on a Common Object Request Broker Architecture (CORBA) technology, includes a set of cooperating components to address the above-mentioned problems. Components appearing as CORBA objects are designed to attain the facility and simplicity of plug-and-play operation. The APC framework integrates with a legacy Manufacturing Execution System (MES) and enables run-to-run control for multiple equipment in semiconductor manufacturing.

The APC Framework performs automatic process control operations through the design and development of a software framework that integrates factory, process, and equipment control systems. The APC Framework benefits semiconductor-manufacturing manufacturing, factories, or "fabs," throughout the development of the APC Framework by using an iterative development approach. The APC Framework is designed to integrate seamlessly with commercially-available APC tools

The APC Framework specifies components and a component structure that enable multiple vendors to build and sell framework-compatible products using an open architecture that accommodates plug-and-play components. The APC Framework advantageously increases product yield distributions and equipment utilization, and lowers defect densities.

Performance specifications of the APC Framework are driven by the requirement for reduced feature size on semiconductor wafers. The APC Framework is integrated with manufacturing tools and a File Control System (FCS). Components of APC Framework are to be commercially or internally supported at some time in the future. APC process control models/algorithms are developed internally. The APC Framework augments existing equipment controllers.

In accordance with an embodiment of the present invention, a computer program product includes a computer usable medium having computable readable code embodied therein including a process control software system controlling a process having a plurality of devices communicating in a network, the devices including a metrology machine, a processing machine, and a controller. The process control software system includes a metrology machine plan routine controlling operations of the metrology machine. The metrology machine plan routine generates a human readable text describing activities to be exercised by the metrology machine and data to be collected and analyzed by the metrology machine. The process control software system also includes a processing machine plan routine controlling operations of the processing machine. The processing machine plan routine generates a human readable text describing activities to be exercised by the processing machine and data to be collected and analyzed by the processing machine. The process control software system further includes a strategy routine controlling operations of the controller. The strategy routine coordinates activities of the metrology machine plan and the processing machine plan that span multiple processing steps of the process.

The illustrative test system and operating method have many advantages. Advantageously, the Object Management Group (OMG) Interface Definition Library (IDL) supports the definition of interfaces. The interfaces are stable so that clients and servers are developed independently. Independent development is best accomplished through usage of OMG IDL in the definition of exceptions, attributes, and sequences.

The OMG IDL-to-C++ advantageously supports standard and alternative mappings for modules. The OMG IDL-to-C++ supports a standard mapping for compilers that support name spaces and nested classes and an alternative mapping for other compilers. The alternative mappings disadvantageously may result in portability problems. Furthermore, OMG IDL advantageously supports a number of basic and constructed data types that are mapped via an IDL compiler into data structures suitable for a particular programming language. For example, in the C++ language data types are limited to fairly primitive types for sequences and strings.



Other data structures are used that offer more functionality and do not impact performance.

## BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

FIG. **1**, labeled PRIOR ART, is a schematic block diagram showing a traditional SPC process.

FIG. **2** is a schematic block diagram showing material flow of a semiconductor manufacturing process from a process engineer perspective.

FIG. **3** is a schematic block diagram showing material flow of a pre-process measurement step of a semiconductor manufacturing process from a process engineer perspective.

FIG. **4** is a schematic block diagram showing material flow of a processing step of a semiconductor manufacturing process from a process engineer perspective.

FIG. **5** is a schematic block diagram showing material flow of a post-process measurement step of a semiconductor manufacturing process from a process engineer perspective.

FIG. **6** is a schematic block diagram showing APC Framework components that support APC scenarios.

FIG. **7** is a schematic block diagram illustrating the architecture of a typical Advanced Process Control (APC) component.

FIG. **8A** is a schematic block diagram illustrating a system architect perspective in a plan startup phase an Advanced Process Control (APC) system for performing an Advanced Process Control (APC) strategy.

FIG. **8B** is a schematic block diagram illustrating a system architect perspective of the APC system using Plug-In Execution.

FIG. **8C** is a schematic block diagram illustrating a system architect perspective of the APC system during a Processing Measurement Step.

FIG. **8D** is a schematic block diagram illustrating a system architect perspective of the APC system during a Post-Processing Measurement Step.

FIG. **9** is a schematic block diagram that illustrates an embodiment of an AddOnSensor Interface (SI) component.

FIG. **10** is a schematic state transition diagram depicting a Data Collector.

FIG. **11** is a class diagram showing an embodiment of the AddOnSensor class.

FIG. **12** is a class diagram showing an embodiment of the communication layer Data Acquisition (DAQ) class.

FIG. **13** is an object collaboration diagram showing an embodiment of the AddOnSensor Initialization class.

FIG. **14** is a collaboration diagram showing an embodiment of the communication layer (DAQ) Initialization.

FIG. **15** is a collaboration diagram showing an embodiment of the communication layer (DAQ) Clean Up.

FIG. **16** is a schematic block diagram showing a process for performing a setup of data collection in the APC.

FIG. **17** is a schematic block diagram showing a process for performing enabling of data collection in the APC.

FIG. **18** is a schematic block diagram showing a process for performing disabling of data collection in the APC.

FIG. **19** is a schematic block diagram showing a process for starting data collection in the APC.

FIG. **20** is a schematic block diagram showing a process for stopping data collection in the APC.

FIG. **21** is a schematic block diagram showing a process for performing data collection buffering in the APC.

FIG. **22** is a schematic block diagram showing a process for performing data collection triggering in the APC.

FIG. **23** is a schematic block diagram showing a process for performing an unsetup of data collection in the APC.

FIG. **24** is a schematic block diagram illustrates an UML collaboration diagram.

The use of the same reference symbols in different drawings indicates similar or identical items.

## DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

The APC Framework includes specification of various project requirements, objectives, and implementation criteria. The APC Framework further defines a list of high-level system functions, a set of system-level use cases, and a data dictionary defining a glossary of terms and concept definitions. The APC Framework also includes specification of a system-level architecture design and identification of subsystems. In the illustrative embodiment, the APC Framework utilizes an Object Modeling Technique (OMT) methodology to set forth the design approach and artifacts generated.

Once the APC Framework subsystems are identified, the APC Framework proceeds through an iterative process of analysis, design, implementation, deployment, and commercialization following a final iteration. In each phase, the APC components are incrementally enhanced in functionality. For each successive iteration, emphasis on component functionality is decreased and replaced by emphasis on applying the framework to solving APC problems of increasing complexity. At the end of an iteration, a comprehensive integration test is performed to ensure that all components work according to the IDL specifications and a performance evaluation is completed before the components are deployed in a fabrication facility.

The iterative process allows rapid evaluation and validation of the APC Framework concepts. Each analysis phase at different iterations allows reevaluation of design philosophy and approach, tools, and methodologies, before further enhancing the system.

In the illustrative embodiment, the APC Framework is implemented in framework components using C++ on Windows NT-based platforms. Some user interface clients are implemented in Java.

Referring to FIG. **2**, a schematic block diagram shows material flow of a semiconductor manufacturing process from a process engineer perspective. The diagram shows how the APC Framework **200** supports a typical run-to-run control scenario. The diagram is presented from the perspective of the framework's primary user, the Process Control Engineer. Concepts including "plan" and "strategy" clarify ideas and crystallize concepts. A "plan" is a human readable text describing activities that are exercised and data that is collected and analyzed. Plans orchestrate all APC activities. A strategy is similar to a higher level plan, but coordinates activities that span multiple processing steps. For example, a strategy specifies the order of running of plans.

The illustrative APC Framework **200** includes a process model **202** that receives feed-forward and feed-back data and calculates a processing parameter. The illustrative portion of the APC Framework **200** includes two measurement devices, in particular a pre-process metrology machine **204** and a post-processing metrology machine **206**. The pre-process metrology machine **204** performs a measurement on a material prior to processing in a processing machine **208** and sends the measurement, as feed-forward data, to the process model **202**. The processing machine **208** sends processed material to the post-processing metrology machine **206** to measure post-process data which is sent to the process model **202** as feedback data.

Plans in the APC Framework **200** include a pre-process metrology plan #1 **210**, a process material plan #2 **212**, and a post-process metrology plan #3 **214** which are defined within the APC strategy #1 **216**.

Referring to FIG. **3**, a schematic block diagram shows material flow of a pre-process measurement step **300** of a semiconductor manufacturing process from a process engineer perspective. An APC plan **302** sends a message to a machine **304** to measure a material. The machine **304** sends measured measurement data back to the plan **302**. The plan **302** stores the measurement data in a data store **306** of a database **308** for usage at a processing step. The plan also sends the measurements to a data history store **310** of a historical database **312**.

Referring to FIG. **4**, a schematic block diagram shows material flow of a processing step **400** of a semiconductor manufacturing process from a process engineer perspective. An APC plan **402** retrieves a process model from the data store **306**, then executes a parameter calculation algorithm **404**. The APC plan **402** gives the calculated parameters to a machine **406** and directs the machine **406** to execute the process. The machine **406** issues a signal to the APC plan **402** when the process execution is complete. The APC plan **402** sends the calculated parameters to the data history store **310** of the historical database **312**.

Referring to FIG. **5**, a schematic block diagram shows material flow of a post-process measurement step **500** of a semiconductor manufacturing process from a process engineer perspective. An APC plan **502** sends a message to a machine **504** instructing the machine **504** to measure a post-processed material. The machine **504** sends measurement data to the APC plan **502**. The APC plan **502** retrieves an old process model from the data store **306**. The APC plan **502** executes a model update algorithm **506**. The APC plan **502** stores an updated model in the data store **306** for usage in the processing step **400**. The APC plan **502** sends new model data to the data history store **310** of the historical database **312**.

Referring to FIG. **6**, a schematic block diagram shows APC Framework components that support APC scenarios. TABLE I summarizes the APC components and the primary functionality of the components.

| Component | Description |
|---|---|
| Execution/Control/ Monitoring Components | |
| Plan Execution | Provides for execution of APC strategies, plans, and associated process control scripts, interacting with other components as dictated by the contents of the scripts to provide desired process control functionality. |
| Fault Detection Monitoring | Provides factory operations and engineering personnel with a "window" into the current and past state of processing equipment, including processing activity, alarms, and faults. |

-continued

| Component | Description |
|---|---|
| Capability Providers | |
| Machine Interface | Provides an interface between MES equipment interfaces (EIs) and the APC representation of a fab tool. Primarily translates between EI-specific communications and APC CORBA. |
| Sensor Interface | Provides the appropriate interface environment to execute sensor data acquisition Plug-in applications (e.g., Labview VI). |
| Operator Interface | Facilitates communication between a wafer fab technician (WFT) and the APC system via a graphical user interface (GUI). |
| Application Interface | Provides the appropriate interface environment to execute control Plug-in applications such as Matlab and Mathematica. |
| Document Management Components | |
| Document Management | Provides base implementation of documents under version control for extended implementation by other Document Management components (Data Collection Plan Management, Plug-in Management, and Plan Management). |
| Data Collection Plan Management | Extends Document Management for configuration and management of data collection plans, associated duration plans, sampling plans, and reporting plans. At run time, provides the appropriate plan to Plan Execution Management component. |
| Plug-in Management | Extends Document Management for definition, importing, and management of process control Plug-in applications that are developed with tools external to the APC system, such as Matlab, Mathematica, MatrixX, etc. |
| Plan Management | Extends Document Management for definition, configuration, and management of APC strategies, plans, and scripts, and defines when they are to be used. At run time, tracks strategy execution progress. |
| Sign-Off Management | Provides change management, sign-on, and effectivity administration to support other Document Management components. |
| Data Storage Components | |
| Data Store | Stores and retrieves control models and status data required for process control. |
| Data History | Provides for historical repository and archival of APC data for use in off-line analysis. |
| Administrative Support Components | |
| Component Management | Provides administrative, configuration, event, and state services for all servers developed for the APC framework. |
| System Management | Defines, groups, installs, and manages the components in the APC system. |
| Logger | Provides centralized services for capturing activity and trace information for diagnostic and monitoring purposes. |
| Registry | Maintains a centralized repository of component configuration information, including setup values, system environment settings, and lists of dependent objects and event channels. |
| CORBA Services Components | |
| Events | Provides basic support for asynchronous events (decoupled event suppliers and consumers), event "fan-in," notification "fan-out," and-through appropriate event channel implementations-reliable event delivery. |
| Trader | Supports a service-based lookup for components to find other components which provide needed services. Component lookups can be constrained to limit the components to be retrieved, based on component-specific or instance-specific properties. |

Referring to FIG. **7**, a schematic block diagram illustrates the system architecture of a typical Advanced Process Control (APC) component **700** and shows functional interconnections and architecture design details common to a plurality of components in the APC framework. The architecture diagrams show scenarios from a system architecture perspective. The architecture diagrams highlight how interactions between APC components achieve the desirable outcome for the Process Engineer. Boxes with rounded corners represent objects, single-headed arrows are method calls, and double-headed arrows are events.

APC components **700** have characteristics and behaviors that are defined by the APC Framework to set a base level of functionality of all components. A plurality of APC components **700** cooperate to perform APC applications such as run-to-run control and have specified common behaviors including behaviors that support installation and system administration.

The APC component architecture defines a single set of services available to all clients. A system management client **702** specifically exploits the services and interacts with various components to install initial versions and new releases of the components and to control the logging levels of the components. An Object Management Group (OMG) Interface Definition Library (IDL) interface is defined to support a base level of component functionality. All components inherit the common OMG IDL interface which sets the fundamental functionality of the component. Functionality of a particular component is further defined to supply an individual domain-specific functionality. The APC framework supplies a single implementation of the common OMG IDL interface so that domain-specific components inherit the OMG IDL behavior without concern about implementation. The common basic shared-implementation of the interface advantageously increases robustness of the system. Although a single implementation of the interface is supplied for usage of the components, individual components may have a unique implementation of the interface, if desired, so long as returned object fully supports the base OMG IDL interface.

The system management's client **702** binds to a selected component **700** at run-time and the component returns an object that supports the common OMG IDL interface. As long as the base interface supports the OMG IDL specifications, the system management client **702** safely guides a returned object to the interface. The system management client **702** can then apply operations defined for the interface to install a component, upgrade a component, and control the logging levels. The common base interface advantageously simplifies the implementation of the system management client **702** across a wide range of components.

The system management client **702** is the component that calls a bind function to obtain object references. Other components, except a trader component, import bindings from a trading service (not shown). The ACP architecture localizes uses of the bind operation because, although supplied by a number of vendors, bind is not a CORBA-

compliant way to obtain the object references. In the case of a system manager server **704**, bind is used to break a cyclic dependency and bootstrap the system. The system manager **702** uses object references to install the components, but does not obtain the object references from the trading service until the components are installed. By using bind, the system manager server **704** obtains and exports the object references to the trader. Once the components are installed, all components can use the trader to import the object references.

The trader is an initial reference that is retrieved in a non-standard fashion using bind. A CORBA **706** initialization service supplies a standard technique to resolve initial references using the Object Request Broker(ORB), but only uses a conforming implementation to support a naming and interface repository. In the illustrative embodiment, the trader is not a mandatory reference that is resolved using the ORB so the ORB is used to resolve an initial reference to the naming service and the naming service is used to find the trader. In other embodiments, the trader service is obtained via a call to bind. The call to the bind function is localized inside a single procedure. After acquiring the trader, other references are resolved either through the trader or by applying operations to objects that are retrieved from the trader. In further additional embodiments, the trader is a mandatory reference.

A profile contains component-specific information that can be specified after the component **700** has been compiled. The profile is loaded at run time by the component **700** when the component is started. The system manager **702** creates the profile as part of the installation of a component. The profile is stored in a registry **708** for later retrieval. At run time, the name of the APC component **700** is passed on a command line (not shown). The APC component **700** uses the component name to access the profile allocated to the component in the registry **708**. The profile includes three sequences: a first sequence containing internal variables, a second sequence containing environment variables, and a third sequence containing the binding variables.

The profile specifies internal variables for component-specific settings. For example, a component may use a first database during testing and a second database during production. In another example, a component may capture timing and tracing information during testing, but not during deployment. Internal settings in a profile are represented as a sequence of name-value pairs where the value is a string.

The profile has environment variable settings that are used, for example, to support operation of components that integrate legacy systems. For example, a CORBA component that acts as an Oracle client **710** has a LIBPATH set to dynamically load Oracle client libraries correctly. Failure to correctly set the LIBPATH results in a runtime exception. The environment variables inside a profile are represented as a sequence of name-value pairs where the value is a string.

The profile has binding variables that specify a selection criterion used at runtime to import service providers to the APC component **700**. In multi-tier architectures, components that use the services of other components are prevalent. For a robust system, APC components **700** do not statically bind to these service providers. Instead, the APC components **700** dynamically acquire service providers at runtime for greater flexibility and robustness. If a requested service provider is inoperable for an extended time, the service provider is selectively replaced by changing the binding information in the profile. A binding variable is also used to locate either a logging service **712** that the APC component **700** uses to send timing and tracing information, or the event channels **714** the APC component **700** uses to send and receive events. Binding variables are stored in the profile as a sequence of name-value pairs where the value is a structure defined in OMG IDL as:

```
struct BindingValue{ string type; string constraint; boolean mandatory_presence; };
struct BindlngVariable{ string name; BindingValue value; };
typedef sequence<BindingVariable >BindingVariableSeq;
```

The mandatory presence field is used because some bindings are optional. For example, components can be coded to operate with or without a particular binding.

One example of the usage optional bindings arises for a APC component **700** that connot connect to the logging service **712** after successfully processing a long-duration operation. Although the operation is successful, the log is not found, possibly because the log is contained on a machine that is temporarily inoperable. Rather than unwisely aborting the operation merely because the logging service is not found, a better solution is to specify an alternative log. For example, the logging utility may be implemented using a smart proxy so that a failure to connect to the logging service is addressed by logging to a local file or logging to a console based on an internal setting. A logging utility using a technique such as a smart proxy allows the binding handle to the logging service to become optional so that the APC component **700** can operate with or without the logging service operative. The binding value for the optional logging service specifies a "false" designation for the mandatory presence field.

In a distributed system, the location of failure for a thread of control is difficult to determine. A list of potential failure locations is reduced by running components on a single machine. However, the logging of some tracing information, including request, reply, and exception events, is valuable. The logging of tracing information advantageously allows the components to leave a trail so the tester can find and diagnose problems.

Performance bottlenecks are isolated in a system by determining round-trip timing information such as the time to issue a request and receive a reply in response to the request. Once roundtrip times are determined, system developers focus on improving the performance of request and response paths with a longest time duration so that the developer is best able to discover ways to optimize the request. A developer may find a request that obtains several object references, opens and closes a database, and creates a process. In such a request, performance is improved by caching references, opening the database once, and dynamically linking new functionality. The requests that extend the longest time are not always optimized, but the tuning system performance is generally improved.

In a typical system, no tools for tracing and timing intercomponent interactions exist so that support for capturing information is to be incorporated into the system. The system is to be able to disable the capturing of information in a manner that does not impact runtime performance when deployed. An advantageous technique for capturing information is to use filters or interceptors that are called by the ORB at key points during the servicing of a request. Such key points include the time of sending a request, the time of receiving a request, and the time of sending a reply. The filters are implemented to log the tracing and timing information to a central server. The filters achieve correct logging of the information without the application programmers having to make explicit calls.

The System Management Component **702** is used to perform system management and configuration on APC components. An entire APC system uses a single System Management component **702**. The System Manager **702** does not have to be present for APC components to continue running. The System Management Component **702** has two subcomponents: the System Manager Server **704** and the System Manager Graphical User Interface (GUI) **703**. The System Manager Server **704** communicates with the APC components directly, whereas the System Manager GUI **703** communicates only with the System Manager Server **704** via CORBA. The System Manager GUI **703** can be replaced by any available GUI implementations (e.g., Java, Virtual Basic, or PowerBuilder) that comply with CORBA, while the business rules and database access are left strictly to the System Manager Server **704**.

A Machine Interface Component **714** is an interface to a single piece of semiconductor processing or metrology equipment **716**. Through the Machine Interface **714**, the APC system delivers processing parameters, controls equipment operation, collects equipment status information, and receives collected data. The Machine Interface **714** bridges the gap between an existing Equipment Interface (EI) **716** and the APC framework. The Machine Interface **714** translates specific selected messages and events on a CORBA bus **718** to an Isis bus **720**, and back to the CORBA bus **718**. The Isis bus **720** is used by the Equipment Interface **716**. The Machine Interface **714** also facilitates the startup and shutdown operations that are initiated by the System Management Component **702**. The Machine Interface **714** makes publicly available a CORBA event channel. All CORBA events generated by the existing Equipment Interface (EI) **716** are made available on the CORBA bus **718**. In the illustrative embodiment, every instance of a Machine Interface **714** is associated with one and only one (CORBA) Plan Execution Manager and one and only one Equipment Interface **716**.

An Operator Interface (OI) **722** component facilitates communication between an operator **724**, for example a Wafer Fab Technician, and the APC system. Through a CORBA interface **726**, the component allows users to display a variety of pop-up dialogs simultaneously on any number of displays **728**. The Operator Interface **722** also supports for the startup/shutdown and logging features used by the System Management Component **702**.

The OI component **722** has a title base attribute that specifies a string common to all pop-ups. Operations that display pop-ups add either static or user-defined suffixes to the title. For example, if the attribute is set to "CVD09," a call to the notify operation would display a pop-up with "Foo Notification" as the title, whereas a call to the query operation would display a pop-up with "CVD09 Query" as the title.

The OI component **722** also has a display list that specifies the displays in which a pop-up could appear. As a parameter to a pop-up operation, the user may specify that a pop-up appear in only a subset of the display list. Additional operations are defined that allow users to add and remove displays from the display list.

The OI component **722** includes a flexible mechanism for displaying information to the operator via pop-ups. A generic pop-up operation allows users to specify the title suffix, the text message, a list of button labels, an optional response string, a priority level, an optional time out, and a list of displays.

In addition to the generic pop-up, several operations are defined that display specialized pop-ups. These include a notify pop-up that displays a message and "OK" button, a query pop-up that displays a message and "Yes" and "No" buttons, and a prompt pop-up that displays a message, response string, and "OK" button.

An announcement operation is defined as a one-way message that displays a simple pop-up with message and "OK" button. Unlike the other pop-ups, the operation does not wait for user dismissal. This operation does not return any information.

After dismissal, all pop-up operations except the announcement pop-up return the button label that was pressed and the particular display that dismissed the pop-up. If the pop-up displayed a response string, the operator's response is returned. If the pop-up had a time out, a boolean is returned indicating if the pop-up was dismissed due to time out. Dismissing a pop-up on one display causes the pop-up to be removed from all other displays on which it was shown.

An Application Interface (AI) component **730** supplies an interface to an external application **732**, usually an application that runs a control algorithm. The AI **730** executes Plug-ins inside an application associated with the AI component **730**. The Plug-in object contains user-supplied code and non-APC data such as algorithmic constants and setup parameters for the application. The Plug-in executes in the manner of a function call. The Plug-in receives Input parameters. When the Plug-in has finished receiving the Input parameters, the Plug-in returns and supplies Output parameters. The AI component **730** translates data transferred between the Plug-in and APC.

A Data Store Component **734** is used to store APC data created during StrategyRuns. The Data Store Component **734** stores two types of information including permanent data that stores values used by the APC system across runs and Temporary data that APC scripts use to store any general values for a particular strategy run. Examples of permanent data include default model parameter and recipe adjustment values. The Data Store Component **734** provides persistent storage for the APC Plan Execution component. The Data Store Component **734** interacts with the Plan Executor **808**, which is described in the discussion of FIG. **8A**, as an only client.

The APC system also includes a Data History Component **736**. When the APC is running, a data storage location is used to record event occurrences. When a run starts, a recipe changes, run data is collected, or a model parameter value changes, all events are logged so that data collected is later retrieved and analyzed. The APC framework presumes that the history data analysis package is outside the system, and thus a relational database is chosen to store historical records. The data history is stored in a relational-friendly way for retrieval efficiency and usability, as opposed to a more object-oriented view.

The APC component **700** is a fundamental building block of the APC Framework which is distributed, object-oriented, and based on the standards of CORBA, CORBA services, and CORBA facilities. Individual APC components **700** are defined to achieve interoperability so that components operate together, substitutability so that components can be replaced or upgraded, extensibility so that component functionality can be extended and specialized, scalability so that the APC Framework is used on a large or small scale, and migratability so that the APC Framework evolves.

The APC Framework is designed to integrate with legacy systems such as existing monolithic shop floor control systems. Furthermore, the APC Framework integrates with existing process modeling tools such as Matlab and Matrix-X.

The system functional requirements are derived primarily to support two typical scenarios including run-to-run control, and fault detection and classification. Run-to-run control is a model-based process control that usually involves more than one piece of processing equipment. In a typical usage scenario, the results of material processing at one piece of equipment are passed, or "fed-forward" to a subsequent manufacturing step, and used to influence the future processing of the same material. In the APC Framework, run-to-run control is performed according to mathematical process models, and a single application accommodates multiple feed-forward and feedback loops.

Fault detection and classification is model-based detection and classification of process equipment problems. Data is collected from a piece of processing equipment and analyzed using an idealized mathematical process model. Results of the analysis are used to detect the occurrence or likelihood of occurrence of an equipment fault, and to determine the type of the fault.

Both run-to-run control and fault detection and classification are supported using a single set of APC Framework components. Alternatively, the support of different types of APC functionality and adaptation to the specifications of a particular APC installation require considerable flexibility from the APC Framework design. Flexibility is achieved by assembling the APC components in different combinations to conform to application specifications.

Referring to FIG. **8**A in conjunction with FIG. **7**, a schematic block diagram illustrates a system architect perspective in a plan startup phase an Advanced Process Control (APC) system for performing an Advanced Process Control (APC) strategy. A Plan Execution Component **802** services a single semiconductor processing machine (the "machine") and metrology devices associated to the machine. The Plan Execution Component **802** retrieves a plan (the "plan") and scripts associated to the plan from a Plan Manager **804** upon receipt of a "run_APC" command from a Manufacturing Execution System (MES) **801**. The Plan Execution Component **802** then obtains a list of capabilities for suitably executing the plan from a trader **805**, sequences the execution of the plan, and generates a report of plan execution. The report of plan execution specifies whether the plan successfully completed and reports errors, if any, resulting from plan execution.

A Plan Execution Component **802** contains one Plan Execution Manager (PEM) **806** that is responsible for the overall management of all plans executed on the assigned semiconductor processing machine and also responsible for metrology devices associated with the machine. The PEM **806** creates a Plan Executor (PE) **808** to execute a running plan **803**. A PE **808** is responsible for a single plan **803**. The PE **808** exists for the life span of the plan **803** and is destroyed after reporting plan **803** completion. A PE **808** executes a "main script" **807** and may execute one or more "event scripts" that are triggered by events. Functions performed by a PE **808** include: (1) Creating a script executor to execute the main script; (2) Allowing a channel for events to be received from different components; (3) Activating event script executors, for example, if triggered by an event; (4) Maintaining a sharable address space for communications between the main script and the event scripts; and (5) Supplying stubs to map calls from the scripts to external objects.

A Plan Executor (PE) **808** under normal working conditions generally interfaces with the PEM **806**, a Data Store Component **810** (see FIG. **8**C), the Plan Manager **804**, a Machine Interface Component **812** (see FIG. **8**B), an AddOnSensor Interface Component (not shown), a Data History Component **816** (FIGS. **8**C and **8**D), a PlugIn Manager **822** (see FIG. **8**B), and various Application Interface Components (not shown). The Plan Executor (PE) **808** controls a plan startup operation determined by several control parameters including a processing context, runtime parameters, development parameters,

The Plan Execution Manager (PEM) **806** manages interface calls to the APC Plan Execution Component (PEC) **802**. The interface calls include:

```
1. executors();                    //returns a list of current executors
2. executors_running_plan();       //returns a list of Plan Executors (defined
                                   below) running the specified plan
3. run_apc();                      //retrieves and executes a plan
```

The PEM **806** also supplies an interface on behalf of each PE **808** associated to the PEM **806**. The interface calls include:

```
1. plan();      // returns the current plan and context run by the executor.
                This is unique since a PE runs only one plan.
2. stop();      // stops execution of the current plan at the next plan step
3. suspend();   // stops execution of the current plan at the next plan step
4. abort();     // stops execution of the current plan at the next plan step
5. resume();    // resumes execution of the current plan at the next plan step
```

The illustrative interface calls may be defined differently for other applications. The depicted interface calls are selected to reflect typical functions and operations performed by a typical PE **808** and PEM **806**. In other embodiments, a PEM **806** may start multiple plans concurrently using multiple PEs **808**.

The Plan Manager (PM) component **804** manages APC Strategies, Plans, and Scripts artifacts. The Plan Manager **804** creates and configures Strategies and Plans. Scripts are assumed to be created by a separate Script Creation Tool. During runtime, the Plan Manager **804** provides the APC System access to the configured Strategies, Plans, and Scripts.

Plans are sequentially configured in a Strategy. Once a Strategy is selected, an executing instance, the StrategyRun, is created. StrategyRun spans the lifetime of executing all plans in a Strategy. Strategies and StrategyRuns are selected by matching them with an MES Context. This context is matched to the appropriate Strategy using a Context-Specification and a set of Context-Matching Rules. A Context-Specification is a list of specifications that match specified parameters from the context. Thus the complete configurations of a Strategy include a context specification. During runtime, the specification is used to retrieve the appropriate Strategy and execute the associated current Plan of the Strategy. Once the Plan and the Scripts contained in the plan have executed, the Plan Manager **804** prepares the StrategyRun to process the next plan.

Referring to FIG. **8**B, a schematic block diagram illustrates a system architect perspective of the APC system using Plug-In Execution. A Plug-in Management Component **822** includes three main objects, a Manager **824**, a Plug-in **826**, and a Blob **828**. The Manager **824** checks Plug-ins **826** and Blobs **828** out of the persistent storage and manages the memory used for Plug-ins **826** and Blobs **828**. The Plug-in **826** is an object that contains user-defined executable code, any nonvolatile data used to run the code such as constants and initial values, and source code for the

executable program. A Blob **828** is an envelope for data in any number of formats and is used to eliminate the need for APC to know the format of the user-defined code or data. A Plug-in **826** typically contains references to several Blobs **828**.

The executable code contained in a Plug-in **826** within a Blob **828** executes in one of many possible environments including Matlab or Xmath environments. The data in other Blobs **828** referenced by a particular Plug-in **826** is formatted suitably for the appropriate environment. A limitation on the number of execution environments available from APC is undesirable so that the Blob **828**, as a generic envelope, is defined to shield the APC from the specific code and data formats of each execution environment. An Application object **825** configures the Blob **828** for usage in the appropriate execution environment. The format of the Blob **828** contents are immaterial to the Manager **824** and the Plug-in **826**.

The Manager **824** calls the constructors and destructors of the Plug-in **826** and Blob **828**, and passes references to Plug-ins **826**, when requested.

Referring to FIG. **8**C, a schematic block diagram illustrates a system architect perspective of the APC system during a Processing Measurement Step. Following startup, the Plan Executor (PE) **808** notifies the Plan Manager **804** that the plan is started and retrieves permanent stored data tags from the data store manager **809** of the Data Store Component **810**. The plan executes a plug-in to calculate new processing parameter values. The PE **808** sends APC data parameters and a start machine signal to the Machine Interface Component **812**. The Machine Interface Component **812** begins operating and sends a "machine started" signal to the PE **808**. When the operation of the Machine Interface Component **812** is complete, the Machine Interface Component **812** sends a "machine finished" notification to the PE **808**. The PE **808** generates and sends a product run record **817** to the data history manager **815** of the Data History Component **816** and sends run data to the run record **817** of the Data History Component **816**. The PE **808** sends a plan completed signal to the Plan Manager **804**. The PE **808** then terminates while the Plan Manager **804** registers that the second plan for the strategy run is complete.

Referring to FIG. **8**D, a schematic block diagram illustrates a system architect perspective of the APC system during a Post-Processing Measurement Step. The Plan Executor (PE) **808** performs a plan startup. Following startup, the Plan Executor (PE) **808** notifies the Plan Manager **804** that the plan is started and sends data collection information to the machine interface **812** including a setup data collection and an observable data collection.

A Data Collection Plan Management Component (DCPlan) **819** describes the data that is collected by each of the machine interface **812**, sensor interface (not shown) and other interfaces that collect data. The Data Collection Plan Manager (DCPM) **820** is the component responsible for the creation and management of DC Plans. The DCPlan is a data structure used exclusively by the AddOnSensor Interface Component (not shown) and the Machine Interface Component **812**. The DCPlan is defined by the Plan Execution Manager (PEM) **806** including specification of the data to be collected from a particular processing equipment and how the data is reported back to the PEM **806**.

The DCPM component includes two main objects: a Manager **820** and the DCPlan **819**. The Manager **820** checks DCPlans out of the persistent storage and manages the memory used for DCPlans. The DCPlan is an object that contains a set of observables to be collected. The DCPlan also describes the capabilities of a data collector to carry out the data collection task stipulated in the DCPlan **819**.

The PE **808** then starts the machine. The Machine Interface Component **812** begins operating and sends a "machine started" signal to the PE **808**. When the operation of the Machine Interface Component **812** is complete, the Machine Interface Component **812** sends a "machine finished" notification to the PE **808**. The Plan Executor (PE) **808** finds permanent stored data tags from the data store manager **809** of the Data Store Component **810** and updates values in the permanent stored data. The PE **808** generates and sends a product run record **817** to the data history manager **815** of the Data History Component **816** and sends run data to the run record **817** of the Data History Component **816**. The PE **808** sends a plan completed signal to the Plan Manager **804**. The PE **808** then terminates while the Plan Manager **804** registers that the plan is complete.

Referring to FIG. **9**, a schematic block diagram illustrates an embodiment of an AddOnSensor Interface (SI) component **900**, which is the proxy for an external sensor **902** attached to any process equipment **904** controlled by the APC framework. The external sensor **902** may be a simple C++ stand-alone program acquiring data off a thermocouple wire **906**, or a full-fledged LabVIEW application acquiring data using multiple transducers (not shown). Data acquisition is typically performed using a combinations of data acquisition boards **908** and signal conditioning modules **910**. The external sensor **902** communicates with the SI **900** in CORBA messages. A communication layer (not shown) has been created both in the form of a Data Link Library (DLL) that is loaded into LabVIEW and in the form of a C++ class template that is used in a C++ stand-alone program. The function of the communication layer, known as DAQ, is to provide external sensors a means of understanding CORBA messages. Hereinafter a DAQ-enabled external sensor **902** is simply referenced as "DAQ **902**" in this document.

The mapping between an SI **900** and a DAQ **902** is one-to-one in which multiple transducers can be connected to one process equipment. As long as the multiple transducers are controlled by a single DAQ **902**, one SI **900** is responsible for facilitating communications between APC framework components and the DAQ **902**. The DAQ **902** is an application program written to acquire observable measurement from the process equipment using transducers. In some embodiments, the DAQ **902** is a simple C++ program or a lull fledge LabVIEW application using signal conditioning and data acquisition boards. The DAQ **902** communicates with CORBA using either a communication layer DLL or a C++ template.

The Plan Executor (PE) **808** shown in FIG. **8**A sends a Data Collection Plan (DCPlan) to the SI **900** before data collection begins at the DAQ **902**. The SI **900** parses information in the DCPlan and forwards only that information pertinent to DAQ **902**, including Duration Plan, Sampling Plan, Reporting Plan, Observable, and Limits. In addition, the SI **900** forwards to the PE **808** the desired data acquired by the DAQ **902** in a predefined format and in a predetermined time interval. The format and time interval are specified in the DCPlan.

Referring to FIG. **10**, a schematic state transition diagram depicts a Data Collector **1000**. The AddOnSensor (SI) Component **900** inherits the IDL interface from the Data Collector **1000**. The Data Collector **1000** includes an Idle state **1002**, a Ready state **1004**, an Enabled state **1006**, and a Collecting state **1008** that performs data acquisition. All states of the Data Collector **1000** are substrates of a component manager running state.

Referring to FIG. **11**, a class diagram shows an embodiment of AddOnSensor class object model. The component object model diagrams use Object Modeling Technique (OMT) notations to represent the object classes and relationships among the object classes.

Referring to FIG. **12**, a class diagram shows an embodiment of the communication layer DAQ class. FIG. **13** through **15** are object collaboration diagrams showing various aspects of communication layer (DAQ) operation.

Referring to FIG. **16** through FIG. **23**, a plurality of schematic block diagrams show various aspects of data collection in the ACP.

AddOnSensor Class and Operation Specifications are described as follows.

The AddOnSensor (SI) Class is inherited from classes APCTypes_CapabilityProvider, APCComponent_BaseManager, and APCDCInterfaces_DataCollector.

Attribute: sensor_id :string

Responsibilities: denotes the identification string one would use to communicate with

SI. The identification string is issued as a return result to setup_data_collection ( ) call.

Attribute: capability :string

Responsibilities: describes the type of external sensor (DAQ) SI is communicating to,

on behalf of other APC components.

Attribute: buffer_seq :APCRunData: :RunDataSequence

Responsibilities: stores the data collected by the external sensor.

A Class DAQController is described as follows:

Operation: trigger_observed

Responsibilities: Informs the DAQ controller when DAQ observed a trigger described

in the Data Collection Plan.

Parameters: trigger_name :string

Return Value: None

Preconditions: DAQ Controller invoked setup_data_collection method on DAQ

Exceptions: SystemException

A Class SIReportingPlan is described as follows:

Attribute: num_item long

Responsibilities: denotes the number of sequence items in the reporting plan.

Attribute: rp :APCDataCollection::ReportingPlan

Responsibilities: The reporting plan data structure.

Operation: num_rp_item

Responsibilities: returns the number of items in the reporting plan data structure.

Parameters: None

Return Value: long

Preconditions: SI is properly setup.

Exceptions: SystemException

Operation: rp_item_name

Responsibilities: returns the name of an item in the reporting plan data structure.

Parameters: item_idx :long

Return Value: string

Preconditions: SI is properly setup.

Exceptions: SystemException

Operation: rp_item_format

Responsibilities: returns the format (data type) of an item in the reporting plan data structure.

Parameters: item_idx :long

Return Value: ItemFormat

Preconditions: SI is properly setup.

Exceptions: SystemException

Operation: rp_item_value

Responsibilities: returns the value of an item in the reporting plan data structure.

Parameters: item_idx :long

Return Value: depends on the item format.

Preconditions: SI is properly setup.

Exceptions: SystemException

The communication layer (DAQ) Class and DAQ Operation Specifications are, as follows:

Attribute: capability :string

Responsibilities: describes the type of external sensor DAQ is providing the CORBA

communication layer for.

Operation: setup_data_collection

Responsibilities: allows the SI to initiate the set up process at the external sensor.

Parameters: dp :DurationPlan, sp :Samplin_ian, owl :ObservablewithLimitsSequence

Return Value: capability :string

Preconditions: Set up has not been performed before.

Exceptions: SystemException

Operation: enable_data_collection

Responsibilities: allows the SI to enable the data collection process at the external sensor.

Parameters: None

Return Value: None

Preconditions: Set up has been performed.

Exceptions: SystemException

Operation: disable_data_collection

Responsibilities: allows the SI to disable the data collection process at the external sensor.

Parameters: None

Return Value: None

Preconditions: Set up has been performed.

Exceptions: SystemException

Operation: start_data_collection

Responsibilities: allows the SI to start the data collection process at the external sensor.

Parameters: None

Return Value: None
Preconditions: Set up has been performed.
Exceptions: SystemException

Operation: stop_data_collection
Responsibilities: allows the SI to stop the data collection process at the external sensor.
Parameters: None
Return Value: None
Preconditions: Set up has been performed.
Exceptions: SystemException

Operation: unsetup_data_collection
Responsibilities: allows the SI to Un-set up data collection configuration at the external sensor, clean up is performed.
Parameters: None
Return Value: None
Preconditions: Set up has been performed.
Exceptions: SystemException

Operation: get_daq_buffer
Responsibilities: allows DAQ controller to retrieve data collected since the same method was called.
Parameters: None
Return Value: DaqBufferSeq
Preconditions: Set up has been performed.
Exceptions: SystemException

Operation: daq_capability
Responsibilities: allows the external sensor to inform DAQ about it's data acquisition capability.
Parameters: capability string
Return Value: None
Preconditions: None

Operation: num_dp_item
Responsibilities: returns the number of items in the duration plan data structure.
Parameters: None
Return Value: long
Preconditions: set up process started.

Operation: dp_item_name
Responsibilities: returns the name of an item in the duration plan data structure.
Parameters: item_idx :long
Return Value: string
Preconditions: set up process started.

Operation: dp_item_format
Responsibilities: returns the format of an item in the duration plan data structure.
Parameters: item_idx :long
Return Value: string
Preconditions: set up process started.
Operation: dp_item_value
Responsibilities: returns the value of an item in the duration plan data structure.
Parameters: item_idx long
Return Value: string
Preconditions: set up process started.

Operation: num_sp_item
Responsibilities: returns the number of items in the sampling plan data structure.
Parameters: None
Return Value: long
Preconditions: set up process started.

Operation: spjtem_name
Responsibilities: returns the name of an item in the sampling plan data structure.
Parameters: item_idx :long
Return Value: string
Preconditions: set up process started.

Operation: spjtem_format
Responsibilities: returns the format of an item in the sampling plan data structure.
Parameters: item_idx :long
Return Value: string
Preconditions: set up process started.

Operation: spjtem_value
Responsibilities: returns the value of an item in the sampling plan data structure.
Parameters: item_idx :long
Return Value: string
Preconditions: set up process started.

Operation: num_observable
Responsibilities: returns the number of observable in the ObservableLimitsSequence
data structure.
Parameters: None
Return Value: long
Preconditions: set up process started.

Operation: observable_name
Responsibilities: returns the name of an observable in the ObservableLimitsSequence data structure.
Parameters: obs_idx :long
Return Value: string
Preconditions: set up process started.

Operation: observable_format
Responsibilities: returns the format of an observable in the ObservableLimitsSequence data structure.
Parameters: obs_idx long
Return Value: ObservableFormat
Preconditions: set up process started.

Operation: num_observable_limit
Responsibilities: returns the number of limits in an observable, in the

ObservableLimitsSequence data structure.

Parameters: obs_idx :long

Return Value: long

Preconditions: set up process started.

Operation: limit_name

Responsibilities: returns the name of a limit in an observable, in the

ObservableLimitsSequence data structure.

Parameters: obs_idx :long, limiLidx :long

Return Value: string

Preconditions: set up process started.

Operation: limit_format

Responsibilities: returns the format of a limit in an observable, in the

ObservableLimitsSequence data structure.

Parameters: obs_idx :long_limiLidx :long

Return Value: LimitFormat

Preconditions: set up process started.

Operation: limit_upper_value

Responsibilities: returns the upper value of a limit in an observable, in the

ObservableLimitsSequence data structure.

Parameters: obs_idx :long, limiLidx :long

Return Value: depends on the limit format

Preconditions: set up process started.

Operation: limit_lower_value

Responsibilities: returns the lower value of a limit in an observable, in the

ObservableLimitsSequence data structure.

Parameters: obs_jdx :long, limiLidx :long

Return Value: depends on the limit format

Preconditions: set up process started.

Operation: daq_setup_complete

Responsibilities: allows the external sensor to inform DAQ the success or failure of the set up process.

Parameters: success boolean

Return Value: None

Preconditions: Set up process has started.

Operation: next_action

Responsibilities: allows the DAQ to relay messages from SI to external sensor. This is required only when external sensor does not provide callback capability.

Parameters: None

Return Value: string

Preconditions: Set up process was successful.

A Data Collector IDL is described as follows:

```
#ifndef_APCDCInterfaces_idl_
#define_APCDCInterfaces_idl_
#include<APCDataCollection.idl>
```



-continued

```
#include<APCRunData.idl>
// The APCDCInterfaces module defines the interface of a Data Collector
module APCDCInterfaces {
//    The DataCollector interface is an abstract interface that is mixed
//    with other interfaces to form a concrete interface. The concrete
//    interface defines the state machine and state attributes for itself
//    and the DataCollector. Its operations raise a system exception if
//    the manager isn't running.
typedef string DataCollectorId;
typedef string DataMoniker;
exception InvalidDCPlan
{
string message;
};
exception DCSetupFailed
{
string message;
};
exception DCUnsetupFailed
{
string message;
};
exception InvalidDCId
{
string message;
};
exception InvalidDCParams
{
string message;
};
exception DCStartFailed
{
string message;
};
exception DCStopFailed
{
string message;
};
exception DCEnableFailed
{
string message;
};
exception DCDisableFailed
{
string message;
};
exception InvalidMoniker
{
string message;
};
exception RetrieveDataFailed
{
string message;
};
interface DataCollector
{
DataCollectorId setup_data_collection(
        in APCDataCollection::DCPlan dc_plan
    ) raises (
        InvalidDCPlan,
        DCSetupFailed,
        APCTypes::SystemException
    );
void unsetup_data_collection (
        in DataCollectorId dc_id
    ) raises (
        InvalidDCId,
        DCUnsetupFailed,
        APCTypes::SystemException
void enable_data_collection(
        in DataCollectorId dc_id
    ) raises (
        InvalidDCId,
        DCEnableFailed,
        APCTypes::SystemException
void disable_data_collection (
        in DataCollectorId dc_id
    ) raises (
        InvalidDCId,
```

-continued

```
        DCDisableFailed,
        APCTypes::SystemException
    );
void start_data_collection (
        in DataCollectorId dc_id
    ) raises (
        InvalidDCId,
        DCStartFailed,
        APCTypes::SystemException
    );
void stop_data_collection (
        in DataCollectorId dc_id
    ) raises (
        InvalidDCId,
        DCStopFailed,
        APCTypes::SystemException
    );
APCRunData::RunDataSequence retrieve_data (
        in DataCollectorId dc_id
        in DataMoniker moniker
    ) raises (
        InvalidDCId,
        InvalidMoniker,
        RetrieveDataFailed,
        APCTypes::SystemException
    );
  };
};
#endif
```

A Data Collector Events IDL is described as follows:

```
#ifndef_APCDCEvents_idl_
#define_APCDCEvents_idl_
#include<APCRunData.idl>
#include<APCDCInterfaces.idl>
module APCDCEvents
{
struct DataAvailable
 {
 APCDCInterfaces::DataCollectorId id;
 APCDCInterfaces::DataMoniker moniker;
 APCRunData::RunDataSequencedata;
 };
 interface DCPushConsumer: CosEventComm::PushConsumer
  {
  oneway void data_available
  (
  in APCDCInterfaces::DataCollectorId id,
  in APCDCInterfaces::DataMoniker moniker,
  in APCRunData::RuriDataSequence data
  };
 };
 interface DCProxyPushConsumer:
 CosEventChannelAdmin::ProxyPushConsumer
 {
 oneway void data_available
 (
 in APCDCInterfaces::DataCollectorId id,
 in APCDCInterfaces::DataMoniker moniker,
 in APCRunData::RunDataSequence data
 );
 };
};
#endif
An AddOnSensor Interface IDL is described, as follows:
    #ifndef_APCAddOnsensor_idl_
    #define_APCAddOnSensor_idl_
    #include<APCDCInterfaces.idl>
    #include<APCRunData.idl>
    // The AddOnSensor interface . . .
    interface APCAddOnSensor
    APCComponent::BaseManager,
    APCTypes::CapabilityProvider,
    APCDCInterfaces::DataCollector
```

-continued

```
    {
    };
    #endif
A DAQ Controller IDL is described as follows:
    #ifndef_APCDAQController_idl_
    #define_APCDAQController_idl_
    #include<APCRunData.idl>
    // The AddOnSensor interface . . .
    interface APCDAQController
    {
    void trigger_observed
    (
        in string trigger_name
    ) raises
    (
      APCTypes::SystemException
    );
    };
    #endif
A DAQIDL is described, as follows:
    #ifndef_APCDataAcquisition_idl_
    #define_APCDataAcquisition_idl_
    #include<APCTypes.idl>
    #include<APCDataCollection.idl>
    #include<APCDAQController.idl>
    //   The DAQ interface . . .
    module APCDataAcquistion
     {
     typedef sequence<float> SensorData;
     struct DaqBuffer
        {
        string name;
        SensorData data_seq;
        APCTypes::Timestamp time_stamp;
        };
     typedef sequence<DaqBuffer > DaqBufferSeq;
     typedef string Capability;
     interface DataAcquisition
        {
        Capability setup_data_collection
            (
            in APCDataCollection::DurationPlan dp,
            in APCDataCollection::SamplingPlan sp,
            in APCDataCollection::ObservableWithLimitsSequence owl
            ) raises
                (
                APCTypes::SystemException
        void unsetup_data_collection
            (
            ) raises
                (
                APCTypes::SystemException
                );
        void start_data_collection
            (
            ) raises
                (
                APCTypes::SystemException
                );
        void stop_data_collection
            (
            ) raises
                (
                APCTypes::SystemException
                );
        void enable_data_collection
            (
            ) raises
                (
                APCTypes::SystemException
                );
        void disable_data_collection
            (
            ) raises
                (
                APCTypes::SystemException
                );
        void set_controller
            (
```

-continued

```
                in APCDAQController controller
            ) raises
                (
                APCTypes::SystemException
                );
        DaqBufferSeq get_daq_buffer
            (
            ) raises
                (
                APCTypes::System_xception
                );
        };
    };
#endif
```

A first task in the testing of distributed CORBA-based systems is a selection of a suitable set of tests. One option is the selection of a single test that tests all components. A more suitable approach is to identify a series of tests beginning with simple tests that exercise a small number of interfaces or components. Additional tests exercise combinations of components and interfaces until the entire system is tested.

Initial tests most advantageously test interactions that involve only a few components such as tests of components that operate solely as servers and, thus, make no calls to other components. Even components that do not interact with other components involve some complexity since multiple aspects are tested to determine correct operational scenarios, and to detect operations out of sequence, state-based tests, bad parameter values, and so on. A single test case is made up of multiple testing scenarios.

Another technique for determining suitable tests of low complexity interactions includes an analysis of a small set of interfaces that are common across a large number of components. For example, all components may provide a common infrastructure to activate component logging, log incoming requests and replies to a central server, and deactivate logging. An implementation of logging interfaces may be deployed in several components and tested against the logging service.

A further technique involves writing of a test case with both normal and abnormal scenarios. An example of an abnormal scenario is a logging test in which the logger fails. A driver program is written to connect to the components and call a common logging interface of the components. After running the test, all components in the system support logging and provide a solid foundation from which to run other tests, since tracing and timing information remain available in the log.

Still another test determination technique involves selection of services that are used by several components in common. One example is selection of an event service, a naming service, a trader, or a registry service. Testing of the interactions between the components and the services is specified early in the process so that early identification of interactions and data values eliminates uncertainty in the design and results in a system that is more likely to meet specifications.

Another test determination technique involves selection of use cases. A scenario is selected for each of the user-initiated events that result in the client sending out remote messages. The tests typically involve several components such as a client, any primary components with which the client interacts, any secondary components with which the primary components interact, and so on. If the client accepts input from the keyboard or mouse, testing tools may be used to record and play back the user inputs to stress test the system for extended periods. Stress tests are useful but utilize a minimum of tracing information to determine what the servers are doing with the request once the request leaves the client.

A further set of interactions to be specified in a test is an end-to-end test that exercises the entire system. In planning the end-to-end test, a tested with a harness for each of the components that participate in the test is used. Without the testbed, an end-to-end test cannot be performed until all components are developed. The testbed advantageously allows a component to be tested as soon as the component becomes available.

Referring to FIG. **24**, a schematic block diagram illustrates an UML collaboration diagram. Collaboration diagrams are diagrammatic tools for analyzing test cases. UML collaboration diagrams specify scenarios that make up a test case and identify the sequencing of interactions among components. Data handled in the UML collaboration diagrams include information for starting and stopping the components in the test, initializing the components' internal state, and identifying the expected inputs and outputs from the test. The information is used to generate harnesses for a specific test scenario, set up a test, and determine whether the test is successful.

Nodes in collaboration diagrams represent components that service incoming requests. The components operate as black boxes that hide internal classes. For integration testing, interactions among internal classes are unimportant and therefore not tested. Interactions among components that are tested. Prior to servicing a request, a component are initialized by possibly creating one or more objects, acquiring references to other objects, and registering with an Object Request Broker (ORB). An initialization section is included with the collaboration diagrams to control component initialization. A component that does not include an initialization section creates objects for each bind request that is received and registers, by component name, with the ORB.

The edges in the UML collaboration diagram identify sequences of interactions and the values that flow from the interactions. The values are encoded using a notation such as an OMG externalization service that is used to represent all OMG data types and exceptions. Many data types and references have a logical and simple encoding that is consistently defined and available at a set-up or initialization time, prior to run-time. Unfortunately, other data types include vendor-specific, opaque data that is confusing and only determined at runtime. An obscure, difficult-to-understand encoding does not create problems if components always return references to a local object. For example, in a simple case, an encoding can specify an expected data type and a returning harness simply creates and returns an object of that type. Unfortunately, the simple case is not always followed. Some systems specify that a component return a reference to an object in another different component. The return of a reference to a different component is allowed by encoding object references using a variable name of the form IOR:<object-id> that has an initialization specified by the test scenario.

FIG. **24** represents an "Order Item" testing scenario that uses a factory **2402** to look up an object and apply methods to the object. The example uses a retail store object **2404**. The test scenario orders an item from the store **2404**. To run the test the retail store object **2404** and the factory **2402** are registered with the ORB. Then the retail store object **2404**, the factory **2402**, and a driver **2406** are started in a console window.

To stop the test, the retail store object **2404** and the factory **2402** are stopped and then uninstalled from the ORB. Documentation of the illustrative test scenario includes data structures, as follows:

1. Factory::bind
   Format: Factory_ptr bind(string)
   Request: {":Factory"}
   Reply: {IOR:factory}

2. Factory::lookup
   Format: Store_ptr lookup(string)
   Request: {":Sears"}
   Reply: {IOR:sears}

2.1. Store::bind
   Format: Store_ptr bind(string)
   Request: {":Sears"}
   Reply: {IOR:sears}

3. Store::prices
   Format: PriceSeq prices( )
   Request: {}
   Reply: {#({"Item_1", 12.99},{"Item_2", 5.99}

4 Store::order
   Format: void order(Item)
   Request: {"Item_1", 12.99,3 }}
   Reply: {}

Problems that span components are detected by reviewing the collaboration diagrams with developers of the components that participate in this test scenario. For example, if components are not threaded and the collaboration diagram shows a loop back to one of the previous components, a deadlock is detected and is repaired before developers write development code.

If a collaboration diagram shows repeated calls from one component to access different data values in the same object, the different data values are encapsulated in a structure and retrieved in a single call. Performance is improved because the number of messages flowing over the network are reduced. If interactions involve a transfer of large amounts of data, performance is improved by encapsulating the data inside an interface and retrieved in chunks rather than all at once. Another option is to pass the name of a file back to the client and let the client use a distributed file system to access the values in the file. Problems, including the illustrative problems and other problems, are efficiently corrected by beginning a process by identifying, specifying, and reviewing test scenarios.

Integration testing includes testing of each scenario of many in sequence until all are tested. Integration testing selectively proceeds from the top down, the bottom up, or a combination.

Top-down testing uses harnesses to test interactions between components in a test scenario. In top-down testing, all the system interactions are tested using harnesses without waiting for implemented components. A component is tested with harnesses when available. The harnesses surround the component under test, so that a component that interacts with other components receives expected results.

Bottom-up testing begins by testing lower-level components, then using lower-level test results to test the higher-level components in a sequence among increasingly higher-level components. Bottom-up testing is preferred for components that are not dependent on other components. A single test program suffices to test independent components. Bottom-up testing reduces implementation of the harnesses for testing alone, since the tested components are used instead of harnesses. If development is staggered so that components become available in time to test components that use the available components, bottom-up testing eliminates a effort expended in developing the harness. Typically, both top-down and bottom-up testing are used.

Execution of an operation is altered by the occurrence of an exception. Before beginning the operation, the exceptions that affect an operation are specified in an "OMG IDL raises" clause. If exceptions are not specified, only OMG standard system exceptions are raised. Rather than relying on system exceptions, exception specifications are advantageously defined even for exceptions that directly correspond to an OMG system exception. Since the Object Request Broker (ORB) can produce system exceptions, if the operation also produces system exceptions then a client process cannot reliably determine the source of a system exception.

Some exceptions are reasonably anticipated although anticipation of all exceptions that different operation implementations generate is difficult or impossible. For example, one can reasonably expect that if an account number is passed into the operation, a corresponding account may not exist so that specification of a NoAccountWithSpecified-Number exception is warranted. In contrast, exceptions such as running out of memory, the ability to communicate with another server, or the inability to open the persistent store are difficult to anticipate and are somewhat implementation-dependent.

The OMG IDL for a system exception is:

```
module Exceptions
  {
  // Enums for the OMGs major codes.
  enum MajorCode{mc_UNKNOWN, mc_BAD_
    PARAM, . . . ,
    mc_OBJECT_NOT_EXIST };
  // system exception
  exception SystemException(MajorCode major_code;
    string error_message;};
  };
```

A guideline for user-defined exceptions includes two aspects: (1) specification of user exceptions (if any) that are anticipated based on the signature of the operation, and (2) a mandatory specification of one catchall exception for the error conditions that are implementation-dependent or cannot be anticipated. The guideline has several advantages. First, by defining user exceptions for anticipated exceptions, additional information is conveyed in the exception, and the exception is easily caught and handled. Second, by defining an all-encompassing system exception, unanticipated exceptions are raised without redefining the IDL. Furthermore, exceptions raised by an exception are differentiated from the ORB system exceptions.

For example, instead of generating the BAD_PARAM system exception to indicate that the application received an invalid parameter, an operation is configured to generate an Exceptions::SystemException, a user exception that contains the same information as the CORBA exception. The Exceptions::SystemException indicates that a bad parameter was passed and includes a message to identify the problem and some possible corrections. Message information associated to an exception is most advantageously supplied using a local file, possibly using an implementation-dependent minor code as a key into the file.

IDL attributes are most advantageously defined through the usage of operations rather than by directly defining the

attributes since attributes are mapped into an operation in the programming language, but the operation cannot raise any user-defined exceptions. The implementation of the operation may be overridden but the generation of a user-defined exception is not valid. For example, if one attempts to set an attribute to an invalid value, the operation generated for an attribute is disadvantageously limited to raise only system exceptions. A better course is define an accessor operation instead of a read-only attribute, and define an accessor and a mutator instead of an attribute.

When creating a new IDL data type, a sequence for the type is most advantageously defined, particularly when new interfaces and top-level structs or unions are defined. Defining a sequence enables the later definition of operation, possibly in other modules, to return a sequence of the defined types. Failure to define a sequence for a type leads to a high probability that IDL changes will be required at a later time.

The OMG standard C++ mapping defines an alternative mapping for modules. Typically, modules are mapped to namespaces or classes so that the types contained in a module are referenced using a "::" notation. For example, if module "Outer" contains a type called "Inner", the code would reference "Outer::Inner." However if the compiler does not support namespaces or nested types, the mapping of modules would use an underscore ("_") notation and the code would be changed to reference all occurrences of the type with "Outer_Inner." Porting the code to different platforms in this case is therefore time-consuming and susceptible to error. To avoid the problem, macros are define that insert the "::" or the "_" between the module and type names. The macros are called when nested types are found. Enumeration literals have a similar problem. Some are nested in the name space (or class), and the alternative mapping defines enumeration literals at global scope.

Instead of storing internal data members as native CORBA types; higher level abstractions such as those found in the Standard Template Library are used for storing and manipulating internal data values. The reason for using higher level abstractions is that the internal data values are to be kept in storage with a copy of the data passed to the Object Request Broker (ORB) upon a request by a client process. Copies of the data are to be made in any case so that little or no overhead is expended in copying the CORBA values into and out of higher level objects. The higher level objects are accessed and manipulated more easily using the higher level abstractions than by manipulating a native CORBA sequence.

Memory-management rules are applied when developing distributed applications using C++. The component manager approach advantageously results in tighter control of core codes. A memory-management analysis tool, called "Purify" performs a strict adherent-to-memory "cleansing" to detect potential memory conflicts and problems such as dangling pointers and unfreed memory in all our components. The Purify tool identifies leaks in libraries or other code beyond the control of the application writers.

In the illustrative embodiment of the APC, applications are supported cross-platform including support of HP/UX and Windows NT platforms. Cross-platform development problems are avoided by selecting tools that support cross-platform use initially. Tools selection is restricted based on the platforms supported by the tools and compatability between tools.

Some embodiments of the APC system utilize load balancing across the multiple process components. Software packages including Orbit-Isis and Orbit-OTM address load balancing. Similarly, some embodiments of the APC system utilize programming packages, such as Orbit-OTS and Orbit-OTM, that improve robustness in a distributed object system.

While the invention has been described with reference to various embodiments, it will be understood that these embodiments are illustrative and that the scope of the invention is not limited to them. Many variations, modifications, additions and improvements of the embodiments described are possible. For example, those skilled in the art will readily implement the steps necessary to provide the structures and methods disclosed herein, and will understand that the process parameters, materials, and dimensions are given by way of example only and can be varied to achieve the desired structure as well as modifications which are within the scope of the invention. Variations and modifications of the embodiments disclosed herein may be made based on the description set forth herein, without departing from the scope and spirit of the invention as set forth in the following claims.

What is claimed is:

**1**. A computer program product comprising:

a computer usable medium having computable readable code embodied therein, the computable readable code including instructions that implement a process control software system capable of controlling a process having a plurality of devices communicating in a network, the devices including a metrology machine, a processing machine, and a controller, the process control software system including:

a metrology machine plan routine capable of controlling operations of the metrology machine, the metrology machine plan routine capable of generating a human readable text describing activities to be exercised by the metrology machine and data to be collected and analyzed by the metrology machine;

a processing machine plan routine capable of controlling operations of the processing machine, the processing machine plan routine capable of generating a human readable text describing activities to be exercised by the processing machine and data to be collected and analyzed by the processing machine; and

a strategy routine controlling operations of the controller, the strategy routine capable of coordinating activities of the metrology machine plan and the processing machine plan that span multiple processing steps of the process.

**2**. A computer program product according to claim **1** wherein:

the metrology machine is a pre-process metrology machine that measures a characteristic of a material prior to supplying the material to the processing machine, the pre-process metrology machine capable of generating a feed-forward measurement data that is communicated from the pre-process metrology machine to the controller; and

the strategy routine utilizes the feed-forward measurement data as an input data to the controller, the strategy routine capable of determining a processing parameter based on the feed-forward measurement data that is applied to the processing machine and determines activities of the processing machine.

**3**. A computer program product according to claim **1** wherein:

the metrology machine is a post-process metrology machine that measures a characteristic of a material

subsequent to processing the material by the processing machine, the post-process metrology machine capable of generating a feed-back measurement data that is communicated from the post-process metrology machine to the controller; and

the strategy routine utilizes the feed-back measurement data as an input data to the controller, the strategy routine capable of determining a processing parameter based on the feed-back measurement data that is applied to the processing machine and capable of determining activities of the processing machine.

**4**. A computer program product according to claim **1** wherein:

the process control software system includes software routines that are:

distributed among the controller, the metrology machine, and the processing machine;

object-oriented; and

based on standards of Common Object Request Broker Architecture (CORBA), CORBA services, and CORBA facilities.

**5**. A computer program product according to claim **1** wherein:

the process devices further include a database;

the process control software system further includes a data store and a data history;

the metrology machine is a pre-process metrology machine that measures a characteristic of a material prior to supplying the material to the processing machine;

the metrology machine plan routine includes:

a routine capable of directing the metrology machine to measure a material;

a routine capable of receiving measurement data from the metrology machine;

a routine capable of storing the measurement data in the data store for use in a processing step; and

a routine capable of sending the measurement data to the data history.

**6**. A computer program product according to claim **1** wherein:

the process devices further include a database;

the process control software system further includes a data store and a data history;

the processing machine plan routine includes:

a routine capable of retrieving a process model from the strategy;

a routine capable of determining a processing parameter based on measurement data received from a metrology machine plan routine;

a routine capable of sending the processing parameter to the processing machine and directing the processing machine to execute a processing activity;

a routine capable of receiving a notification that the processing activity of the processing machine is complete; and

a routine capable of sending determined parameters to the data history.

**7**. A computer program product according to claim **1** wherein:

the process devices further include a database;

the process control software system further includes a data store and a data history;

the metrology machine is a post-process metrology machine that measures a characteristic of a material subsequent to processing the material by the processing machine;

the metrology machine plan routine includes:

a routine capable of directing the metrology machine to measure a material;

a routine capable of receiving measurement data from the metrology machine;

a routine capable of retrieving an old version of a process plan;

a routine capable of executing a model update algorithm;

a routine capable of storing the updated model in the data store for use in a processing step; and

a routine capable of sending the updated model data to the data history.

**8**. A computer program product according to claim **1** wherein:

the devices include a plurality of processing equipment devices; and

the process control software system supports a run-to-run control scenario using model-based process control operating a plurality of the processing equipment devices, a result of material processing at a processing equipment device being passed on to a subsequent manufacturing step using feed-forward control and being used to influence future processing of the material.

**9**. A computer program product according to claim **1** wherein:

the devices include a plurality of processing equipment devices; and

the process control software system supports a fault detection and classification scenario using model-based detection and classification of problems occurring with a processing equipment device, data being collected from a processing equipment device being collected and analyzed using an idealized mathematical process model, a result of the analysis being used to detect an occurrence of a processing equipment device fault and to determine a type of the processing equipment device fault.

**10**. A computer program product according to claim **1** wherein:

the computer usable medium is a communication signal transmitted over a communication channel.

**11**. A computer program product comprising:

a computer usable medium having computable readable code embodied therein, the computable readable code including instructions that implement a process control software system capable of controlling a process having a controller and a plurality of processing equipment devices and metrology machine devices communicating in a network, the process control software system including:

a plurality of process control framework components capable of controlling activities exercised by the processing equipment devices and controlling data collected and analyzed by the metrology machine devices; and

the process control framework components being developed by an iterative process of a plurality of phases including analysis, design, implementation and deployment phases, the process control framework components being incrementally enhanced and having functionality increased in the plurality of phases.

**12**. A computer program product according to claim **11** wherein:

the process control framework components include a plan executor capable of controlling operations of a device such as the processing equipment devices and the metrology machine devices, the plan executor capable of generating a human readable text describing activities to be exercised by the device and data to be collected and analyzed by the device; and

the process control framework components implement a strategy coordinating activities of the plurality of devices that span multiple processing steps of the process.

**13**. A computer program product according to claim **11** wherein:

the process control framework components are interoperable by a user using a user interface, the user performing coordinating activities for operating the plurality of devices.

**14**. A computer program product according to claim **11** wherein:

the process control framework components are substitutable by a user using a user interface, the process control framework components being replacable or upgradable.

**15**. A computer program product according to claim **11** wherein:

the process control framework components are extensible by a user using a user interface, the process control framework components having a functionality that is extended to perform additional activities and specialized for special activities.

**16**. A computer program product according to claim **11** wherein:

the process control framework components are software routines that are:

distributed among the plurality of devices;

object-oriented; and

based on standards of Common Object Request Broker Architecture (CORBA), CORBA services, and CORBA facilities.

**17**. A computer program product according to claim **11** wherein:

the devices include a plurality of processing equipment devices; and

the process control software system supports a run-to-run control scenario using model-based process control operating a plurality of the processing equipment devices, a result of material processing at a processing equipment device being passed on to a subsequent manufacturing step using feed-forward control and being used to influence future processing of the material.

**18**. A computer program product according to claim **11** wherein:

the devices include a plurality of processing equipment devices; and

the process control software system supports a fault detection and classification scenario using model-based detection and classification of problems occurring with a processing equipment device, data being collected from a processing equipment device being collected and analyzed using an idealized mathematical process model, a result of the analysis being used to detect an occurrence of a processing equipment device fault and to determine a type of the processing equipment device fault.

**19**. A computer program product according to claim **11** wherein:

the process control framework components include:

a plan execution component capable of controlling execution of advanced processing control strategies, plans, and process control scripts associated with the control strategies and plans, the plan execution component capable of interacting with other components of the process control framework components as dictated by the scripts to perform selected process control functionalities.

**20**. A computer program product according to claim **11** wherein:

the process control framework components include:

a fault detection monitoring component capable of supplying an information window into a current state and past states of processing equipment, information in the window including processing activity, alarms, and faults.

**21**. A computer program product according to claim **11** wherein:

the process control framework components include:

a machine interface component for interfacing between an equipment interface and a process control representation of a fab tool, and for translating between equipment interface communications and a Common Object Request Broker Architecture (CORBA).

**22**. A computer program product according to claim **11** wherein:

the process control framework components include:

a sensor interface component for interfacing of sensor data acquisition Plug-in applications.

**23**. A computer program product according to claim **11** wherein:

the process control framework components include:

an operator interface component for communicating between a wafer fab technician (WFT) and the process control system via a graphical user interface (GUI).

**24**. A computer program product according to claim **11** wherein:

the process control framework components include:

a Document Management component for executing version control operations for extended implementation by associated Document Management components including Data Collection Plan Management, Plug-in Management, and Plan Management.

**25**. A computer program product according to claim **11** wherein:

the process control framework components include:

a Data Collection Plan Management component for configuring and managing data collection plans, associated duration plans, sampling plans, and reporting plans.

**26**. A computer program product according to claim **11** wherein:

the process control framework components include:

a Plug-In Management component for defining, importing, and managing process control Plug-In applications that are developed with tools that are external to the process control system, such as Matlab, Mathematica, and MatrixX.

**27**. A computer program product according to claim **11** wherein:

the process control framework components include:

a Plan Management component for defining, configuring, managing, and defining usage of process control strategies, plans, and scripts.

28. A computer program product according to claim 11 wherein:

the process control framework components include:

a Sign-Off Management component for executing chance management, sign-off operations, and supporting other Document Management components.

29. A computer program product according to claim 11 wherein:

the process control framework components include:

a Data Store component for storing and retrieving process control models and process control status data.

30. A computer program product according to claim 11 wherein:

the process control framework components include:

a Data History component for storing an historical repository and archival of process control data for usage in off-line analysis.

31. A computer program product according to claim 11 wherein:

the process control framework components include:

a Component Management component for executing administrative services, configuration services, event services, and state services for servers developed for the process control framework.

32. A computer program product according to claim 11 wherein:

the process control framework components include:

a System Management component for defining, grouping, installing, and managing components in the process control system.

33. A computer program product according to claim 11 wherein:

the process control framework components include:

a Logger component for capturing activity and trace information for diagnostic and monitoring operations.

34. A computer program product according to claim 11 wherein:

the process control framework components include:

a Registry component for maintaining a centralized repository of component configuration information including setup values, system environment settings, and lists of dependent objects and event channels.

35. A computer program product according to claim 11 wherein:

the process control framework components include:

an Events component for handling asynchronous event signals including receiving event signals from event suppliers and sending, event signals to event consumers that are decoupled from the event suppliers, the Events component supporting event "fan-in" and notification "fan-out".

36. A computer program product according to claim 11 wherein:

the process control framework components include:

a Trader component for handling service-based lookup for components to find other components that perform a selected service.

37. A computer program product according to claim 11 wherein:

computer usable medium is a communication signal transmitted over a communication channel.

* * * * *

2. Kee et al (U.S. Pat. No. 5,583,780)

US005583780A

**United States Patent** [19]

**Kee et al.**

[11] **Patent Number: 5,583,780**

[45] **Date of Patent: Dec. 10, 1996**

[54] **METHOD AND DEVICE FOR PREDICTING WAVELENGTH DEPENDENT RADIATION INFLUENCES IN THERMAL SYSTEMS**

[76] Inventors: **Robert J. Kee**, 864 Lucille St., Livermore, Calif. 94550; **Aili Ting**, 7329 Stonedale Dr., Pleasanton, Calif. 94558

[21] Appl. No.: **366,579**

[22] Filed: **Dec. 30, 1994**

[51] **Int. Cl.**[6] ..... **G06F 19/00**

[52] **U.S. Cl.** ..... **364/468.24**; 364/149

[58] **Field of Search** ..... 364/149, 468

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 4,796,194 | 1/1989 | Atherton | 364/468 |
| 5,105,362 | 4/1992 | Kotani | 364/468 |
| 5,111,404 | 5/1992 | Kotani | 364/468 |
| 5,305,221 | 4/1994 | Atherton | 364/468 |
| 5,347,460 | 9/1994 | Gifford et al. | 364/468 |
| 5,408,405 | 4/1995 | Mozumder et al. | 364/151 |

OTHER PUBLICATIONS

Peter Singer, Rapid Thermal Processing: A Progress Report, Semiconductor International, May 1993, pp. 64–69.

Laura Peters, The Hottest Topic in RTP, Semiconductor International, Aug. 1991, pp. 56–62.

Lie, H., et al., "Simulation of Thermal Thermal Processing Equipment and Processes", Rapid Thermal Integrated Processing II, Materials Research Society Proceedings; 303, 197–209 (1993).

Ting, A, "The Influence of Wavelength–Dependent Radiation in Simulation of Lamp–Heated Rapid Thermal Processing Systems," 2nd International Rapid Thermal processing Conference, RTP '94, Round Rock, TX, 102–109 (1994).

Kee, et al., "A Differential/Algebraic Equation Formulation of the Method–of–Lines Solution to System of Partial Differential Equation", Sandia National Laboratories, SAND 86–8893 (1986).

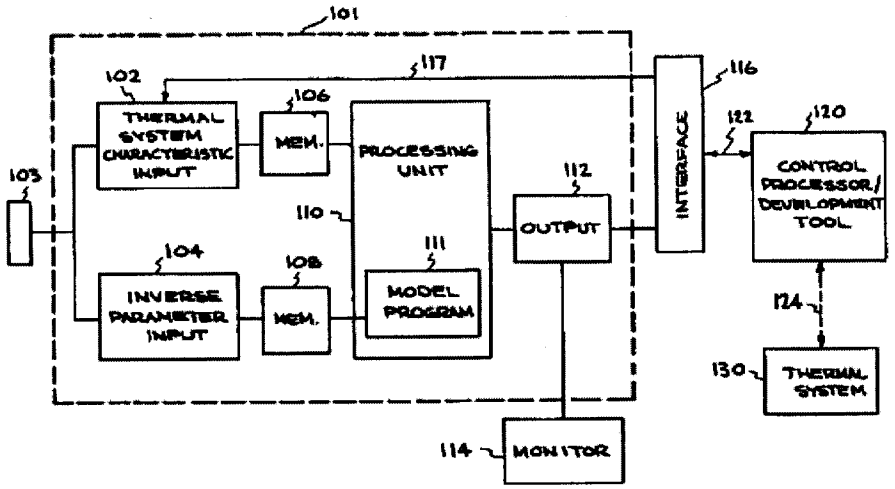
Petzold, "A Description of a DASSL: A Differential/Algebraic System Solves", Sandia National Laboratories, SAND 82–8639 (1982).

*Primary Examiner*—Roy N. Envall, Jr.
*Assistant Examiner*—Y. Kundupoglu
*Attorney, Agent, or Firm*—Donald A. Nissen; Timothy D. Stanley; Gregory A. Cone

[57] **ABSTRACT**

A method and apparatus for predicting the spectral (wavelength-dependent) radiation transport in thermal systems including interaction by the radiation with partially transmitting medium. The predicted model of the thermal system is used to design and control the thermal system. The predictions are well suited to be implemented in design and control of rapid thermal processing (RTP) reactors. The method involves generating a spectral thermal radiation transport model of an RTP reactor. The method also involves specifying a desired wafer time dependent temperature profile. The method further involves calculating an inverse of the generated model using the desired wafer time dependent temperature to determine heating element parameters required to produce the desired profile. The method also involves controlling the heating elements of the RTP reactor in accordance with the heating element parameters to heat the wafer in accordance with the desired profile.

**13 Claims, 7 Drawing Sheets**



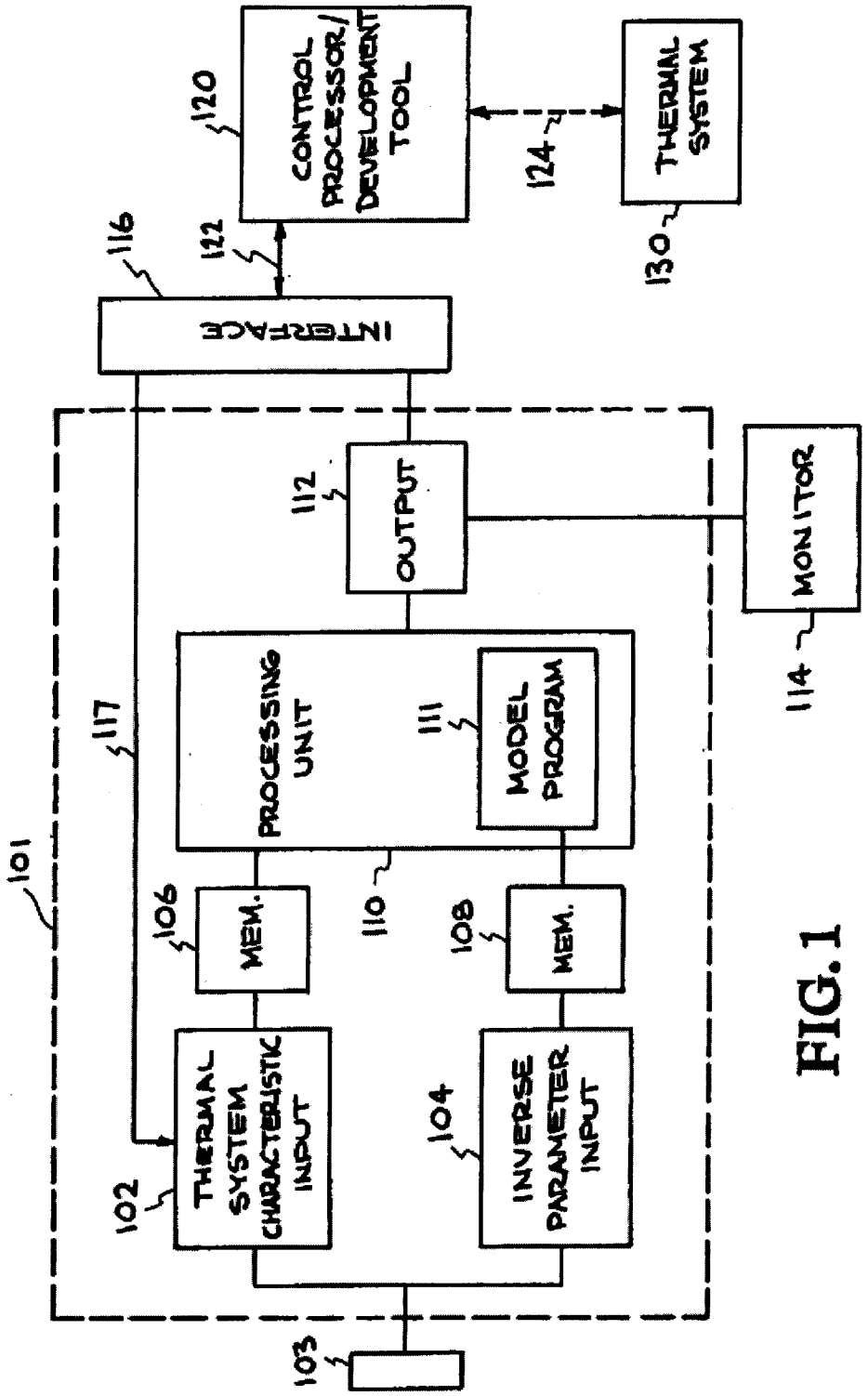

CONTROL PROCESSOR/ DEVELOPMENT TOOL
120
124
THERMAL SYSTEM
130
116
122
INTERFACE
112
OUTPUT
MONITOR
114
117
PROCESSING UNIT
111
MODEL PROGRAM
101
106
MEM.
110
108
MEM.
THERMAL SYSTEM CHARACTERISTIC INPUT
102
104
INVERSE PARAMETER INPUT
103

FIG. 1

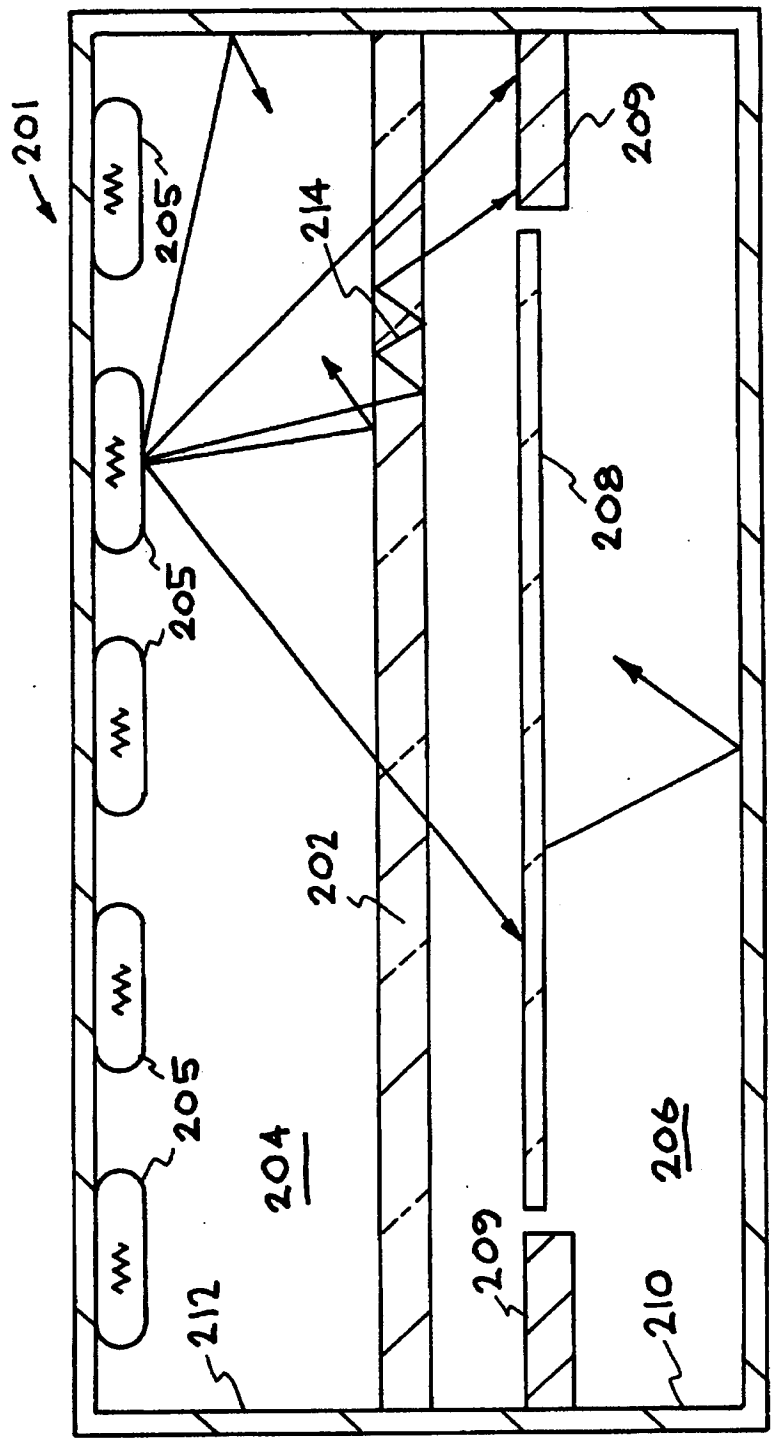



FIG. 2

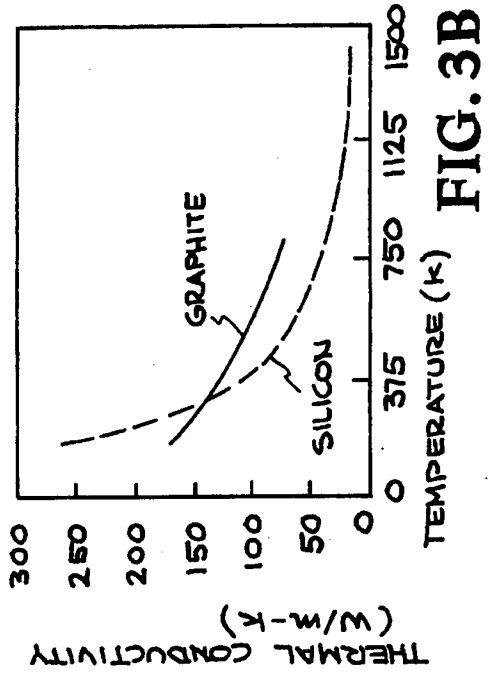



FIG. 3B

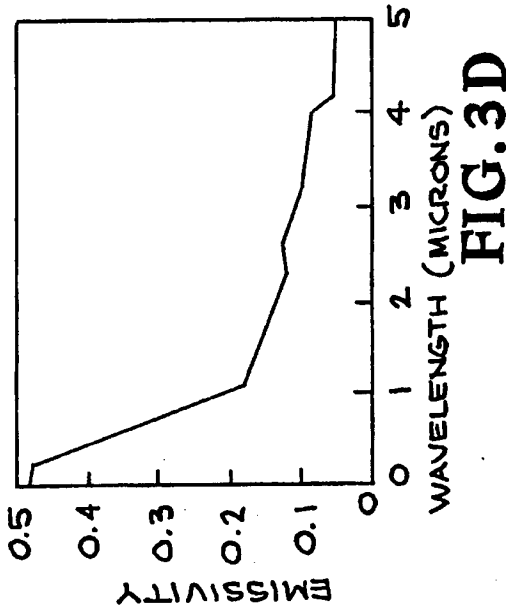



FIG. 3D

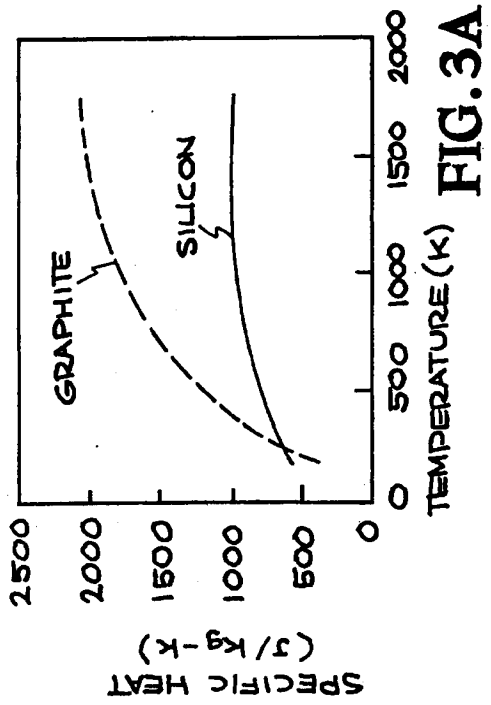



FIG. 3A

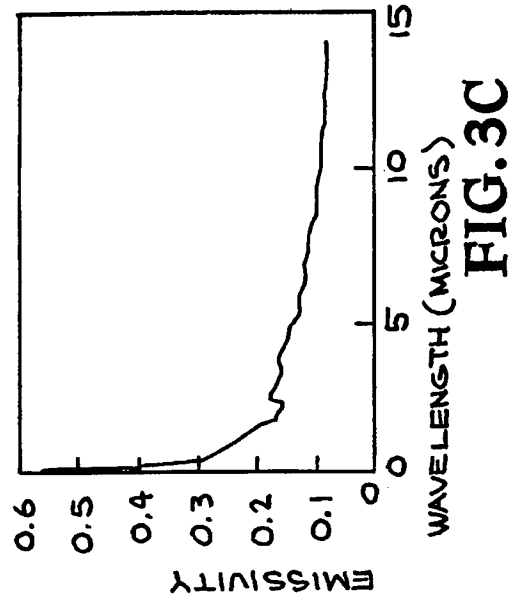



FIG. 3C

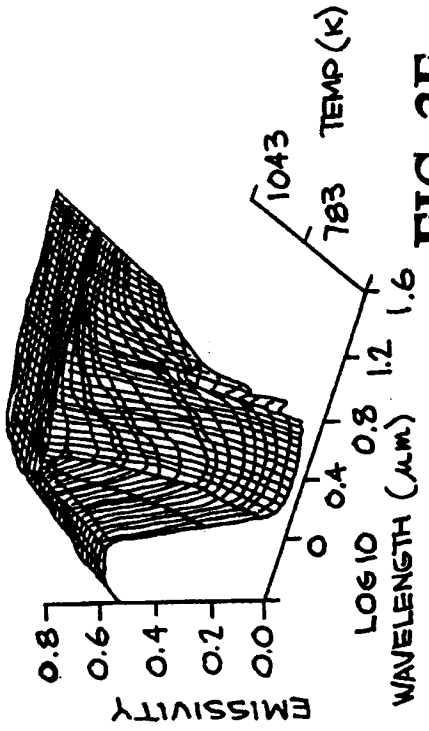



FIG. 3F

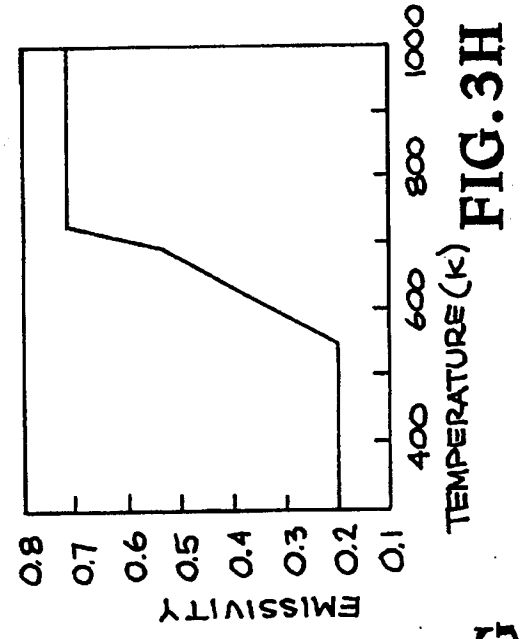



FIG. 3H

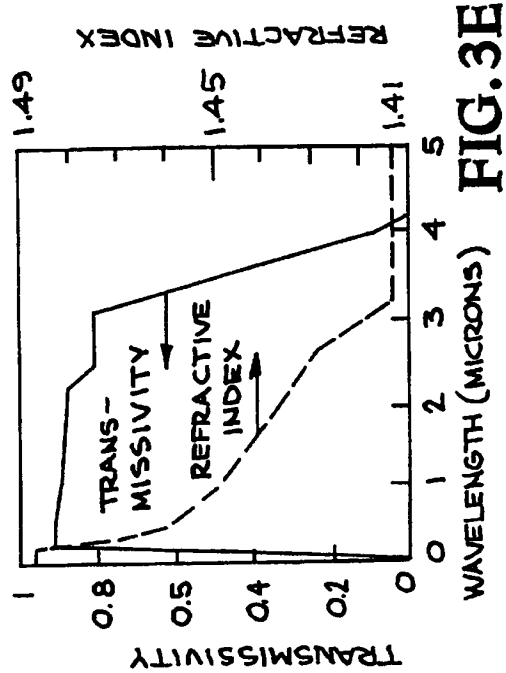



FIG. 3E

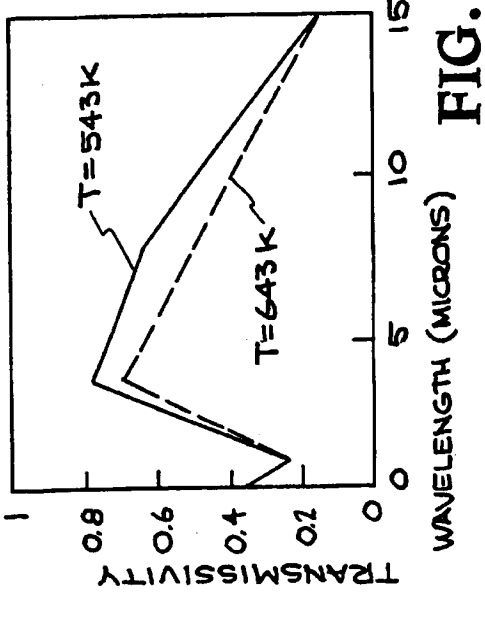



FIG. 3G

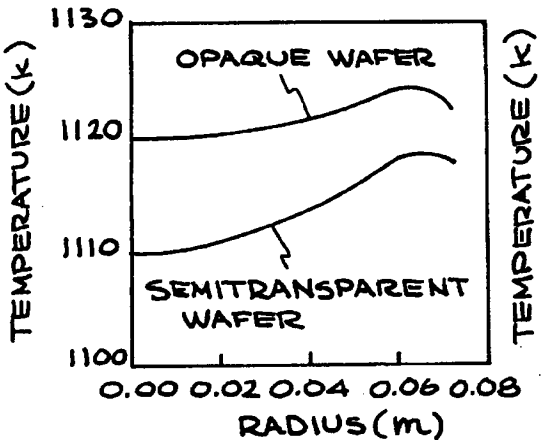



FIG. 5A

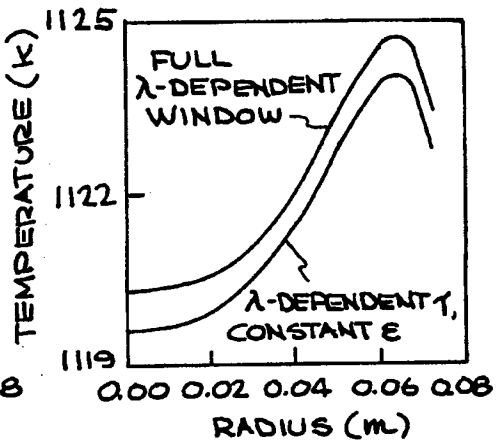



FIG. 5B

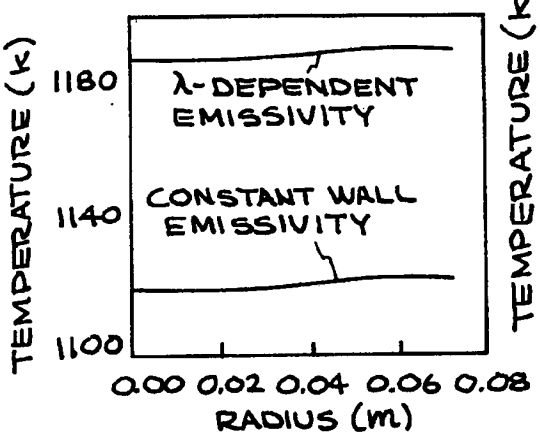



FIG. 5C

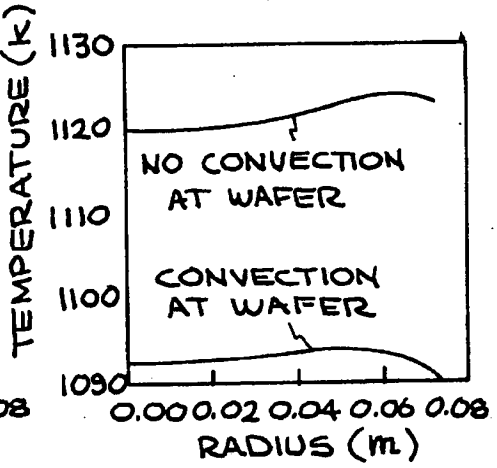



FIG. 5D

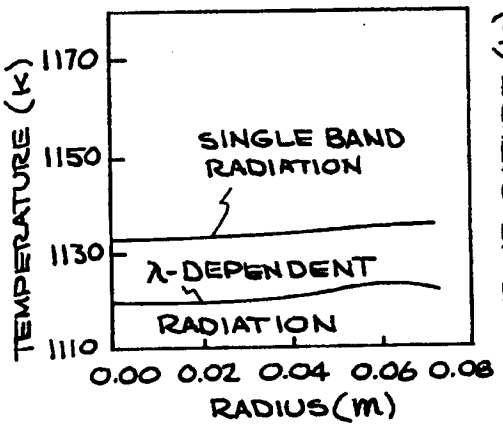



FIG.6A

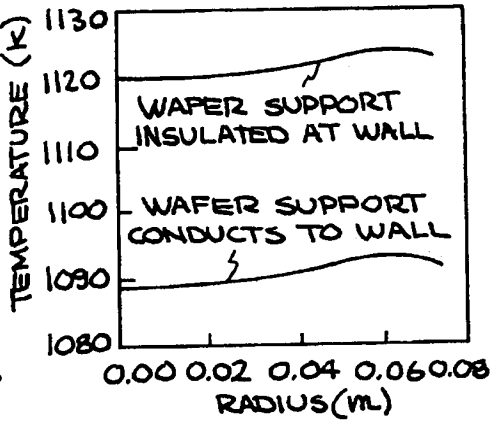



FIG.6B

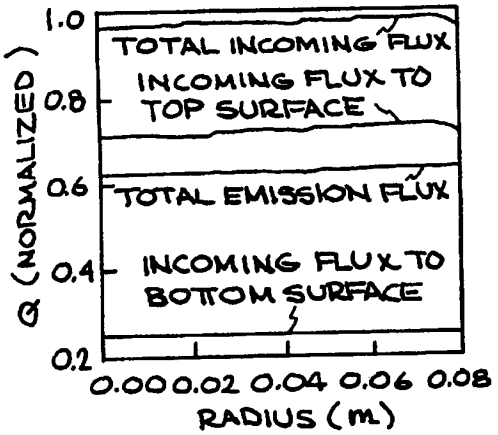



FIG. 6C

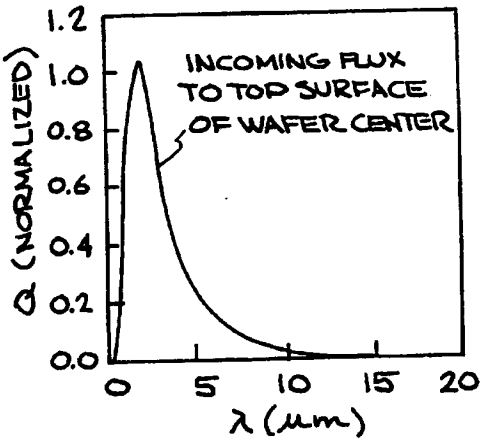



FIG. 6D

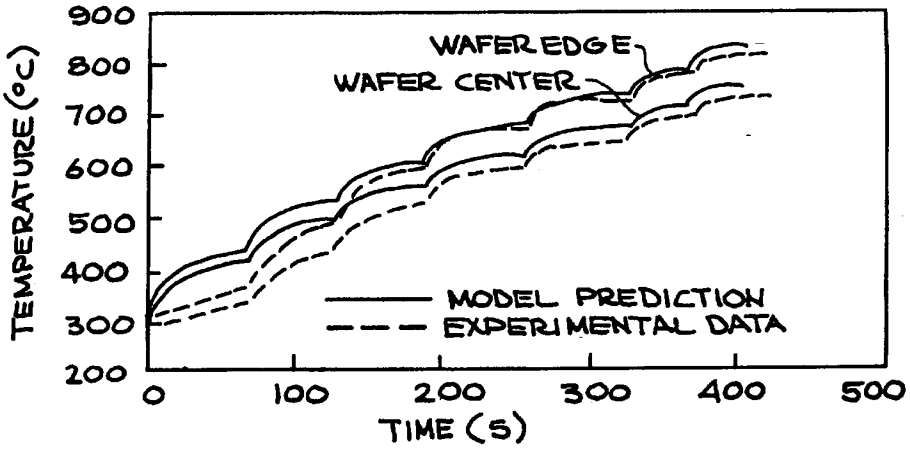

TEMPERATURE (°C)
900
800
700
600
500
400
300
200
0
100
200
300
400
500
TIME (S)
WAFER EDGE
WAFER CENTER
MODEL PREDICTION
EXPERIMENTAL DATA

FIG. 4

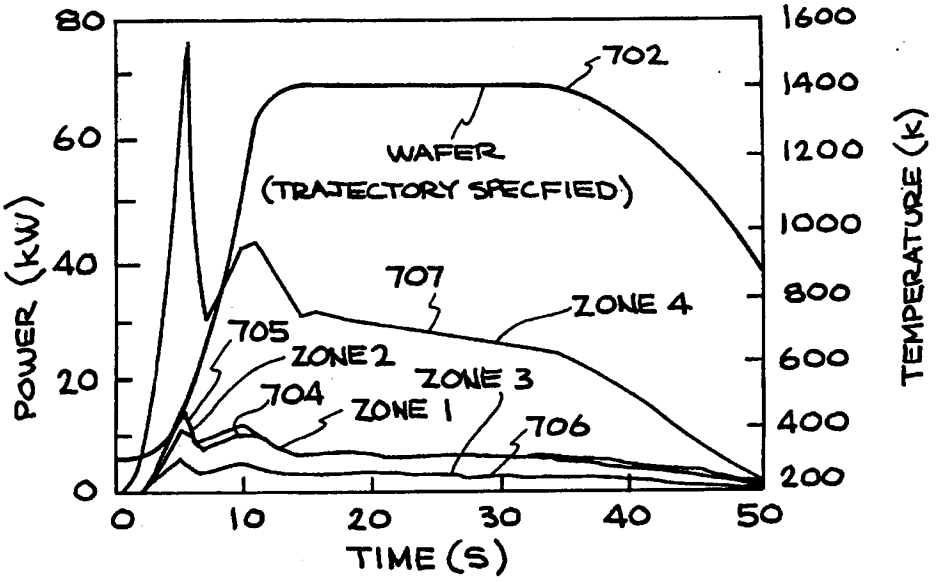

POWER (kW)
80
60
40
20
0
TEMPERATURE (k)
1600
1400
1200
1000
800
600
400
200
0
10
20
30
40
50
TIME (S)
702
WAFER
(TRAJECTORY SPECFIED)
707
ZONE 4
705
ZONE 2
ZONE 3
704
ZONE 1
706

FIG. 7

# METHOD AND DEVICE FOR PREDICTING WAVELENGTH DEPENDENT RADIATION INFLUENCES IN THERMAL SYSTEMS

## BACKGROUND OF THE INVENTION

The instant invention is directed to a method and apparatus for predicting wavelength dependent radiation influences in thermal systems, and more particularly for predicting such influences in systems including semitransparent mediums in a relatively simply manner such that the predictions can be rapidly carried out. The predictive system includes a model which may be executed quickly on workstation class computing platforms and may be used, for example, in the design and control of rapid thermal processing (RTP) systems by permitting rapid comparison of alternative reactor designs and ,physical models as well as real-time model-based control procedures.

Prior attempts to simulate complex thermal systems such as RTP reactors have generally followed two approaches. In a first approach two-dimensional and three-dimensional finite-element or finite-volume approaches have been used to model the heat transport and coupled fluid flow in the reaction chamber. In such systems, the radiation exchange is handled through radiation view or exchange factors. Such approaches generally consider only gray-diffuse view factors. One approach, described in Lie et al., "Simulation of Thermal Processing Equipment and Processes," *Rapid Thermal and Integrated Processing II, Material Research Society Symposium Proceeding*, 303 (1993), also considers specular reflection in their exchange factors. Gray-diffuse view factors are calculated by geometry. Typically, the specular exchange factors are calculated using Monte-Carlo, ray-tracing software. Such a modeling technique is disadvantageous in that it is complex and requires a relatively long time to run.

The second approach to RTP modeling is based on process control. In these efforts, the models consider only one-dimensional conduction in the wafer, with convective and radiative heat fluxes considered at the wafer surfaces. The radiation is computed in terms of geometric gray-diffuse view factors that couple the lamp system to the wafer. While such models may be run relatively quick on a computer, and are therefore suitable for use in designing the real-time control algorithms, these approaches do not take into account non-gray wavelength-dependent radiation.

Because radiation is an electromagnetic wave, an accurate model of radiative energy transport must include the influences of spectral (wavelength-dependent) and directional factors. The conventional technique treats radiative heat transfer between surfaces as diffuse reflectors and diffuse emitters-absorbers. In this way, the problem of diffuse surface interchange is reduced to a geometric problem of determining the view factors between all interacting surfaces, and the directional effects are eliminated.

The values of surface radiation properties (emissivity, transmissivity, and reflectivity) depend on the wavelength, thermodynamic state (e.g., temperature and composition or mole numbers) and the morphology of the surface (e.g., surface roughness). In other words, the radiation emitted, reflected, transmitted and absorbed by a solid body depends on the radiative properties of the material, temperature of the body, viewing direction and the surface conditions.

Typically, conventional attempts to solve the complexity of the radiative transport consider only wavelength-independent radiation (gray-diffuse). Alternatively, some kind of



indirect way to estimate roughly the wavelength-dependent effects is employed. These approaches are not satisfactory, since spectral emissivity plays an important role in the behavior of radiative heat exchange between materials.

## SUMMARY OF THE INVENTION

It is therefore an object of the instant invention to provide a method and apparatus for determining, that is, accurately modeling or predicting, wavelength dependent radiation influences in thermal systems in a manner which can be implemented sufficiently rapidly to overcome the above mentioned drawbacks.

Still another object of the instant invention is to use the method and apparatus of the instant invention to more accurately and rapidly design and control rapid thermal processing (RTP) systems by permitting fast comparison of alternative reactor designs and physical models as well as real-time model-based control procedures.

To accomplish these and other objects of the invention, there is provided a system which includes a modeling apparatus for accurately characterizing time dependent spectral thermal radiation transport of a thermal system. The thermal system has a first portion having one or more heating elements, and a second portion separated from the first portion by a partially transmitting medium. The modeling apparatus includes: an input device for inputting characteristics of the thermal system; a memory connected to the input device for storing the thermal system characteristics, the thermal system characteristics including at least geometric parameters of the thermal system, a time dependent intensity profile of the heating elements and wavelength-dependent properties of the partially transmitting medium; a processing unit connected to the memory, the processing unit predicting a time dependant spectral thermal radiation transport characteristic of the thermal system based on the thermal system characteristics; and means for producing a characteristic model of the thermal system using the time dependant spectral thermal radiation transport characteristic.

In accordance with another embodiment of the invention, there is provided a method of controlling a rapid thermal processing (RTP) reactor for processing a silicon wafer under a desired wafer time dependent temperature profile. The RTP reactor includes a heating element and a partially transmitting component separating the heating element from the wafer. The controlling method includes the steps of: generating in a computer a spectral thermal radiation transport model of the RTP reactor for given heating element parameters; inputting into the computer data specifying the desired wafer time dependent temperature profile; calculating an inverse of the generated model using the data specifying the desired wafer time dependent temperature profile to determine heating element parameters required to produce the desired wafer time dependent temperature profile; and controlling the heating elements of the RTP reactor in accordance with the heating element parameters to heat the wafer in accordance with the desired wafer time dependent temperature profile.

In accordance with still another embodiment of the invention there is provided a method of designing a rapid thermal processing (RTP) reactor for processing a silicon wafer which will include at least one lamp heating element and a partially transmitting component separating the lamp heating element and the wafer. The designing method includes the steps of inputting data into a computer corresponding to

a general design of the RTP reactor including at least the thermal transport characteristic of the partially transmitting component and the wafer and the general design of the thermal radiation characteristic of the lamp heating element; generating in the computer a model of spectral thermal radiation transport of the RTP reactor based on the input data; and selecting components for the RTP reactor based on the model.

## BRIEF DESCRIPTION OF THE DRAWINGS

The instant invention will be better understood from the following detailed description of embodiments of the invention in connection with the accompanying drawings, in which:

FIG. **1** illustrates an apparatus in accordance with an embodiment of the instant invention;

FIG. **2** illustrates a rapid thermal processing (RTP) reactor;

FIGS. **3A–3H** illustrate effects of assumptions made in input parameters used by the apparatus of FIG. **1**;

FIG. **4** is a graph illustrating the results of the apparatus of the instant invention compared with actual experimental data;

FIGS. **5A**, **5B**, **5C**, and **5D** are illustrated sensitivity to physical approximations in the apparatus and method of the instant invention;

FIGS. **6A–6D** illustrate sensitivity to additional physical approximations in the apparatus and method of the instant invention; and

FIG. **7** illustrates an output of an inverse analysis of the instant invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In accordance with the instant invention, a physical model of wavelength dependent radiation transport in a thermal system having particular characteristics is generated in a modeling apparatus, e.g., in a workstation or other computing platform. An apparatus according to an embodiment of the instant invention is depicted in FIG. **1**. In FIG. **1**, a modeling apparatus **101** is generally shown. As more fully described below, the modeling apparatus **101** generates a model of a thermal system sufficiently rapidly to permit the model to be used in design of the thermal system as well as in the development of control software for the thermal system using real-time feed back from the modeling apparatus **101**. In other words, the modeling apparatus **101** may be used to develop a model of the thermal system **130** prior to construction, and the design may be tested using the modeling apparatus **101** for desired performance characteristics. Since the results of the modeling apparatus **101** can be generated very fast, many alternative design parameters may be considered in designing the system in a relatively short period of time. Because the modeling apparatus **101** considers the wavelength dependence of the radiation transport of the thermal system, the modeling apparatus **101** can be used to accurately construct systems requiring stringent temperature control. This is particularly useful in systems where the thermal system includes opaque and partially opaque elements interacting with the thermal radiation.

The modeling apparatus **101** has a thermal system characteristic input section **102** for receiving input characteristics of a thermal system to be modeled from an input device **103** which may include, for example, a keyboard, a mouse,



an interface to another computer platform, etc. The thermal system characteristics are stored in a memory **106** under control of the thermal system characteristic input section **102**. A processing unit **110** (e.g. a CPU) connected to the memory **106**, uses the characteristics about the thermal system stored in memory **106** to predict the operation of the thermal system including the wave-length dependent thermal radiation transport in accordance with a modeling program **111**. The operation of this system and the manner in which it develops a wavelength dependent thermal transport model will be further explained in connection with a more specific example of an RTP system provided below.

The predictive model generated by the modeling apparatus **101** is provided to an output section **112**. The output section **112** may be connected to a monitor **114**, for example, to display the results. Alternatively, the output could be provided to any number of output devices, such as a printer, plotter, etc. These results may be used in the design of a thermal system directly by comparing the model generated by the modeling apparatus **101** with a desired function or characteristic of the thermal system to be built.

The modeling apparatus **101** of the instant invention may also be used to perform an inverse analysis to establish the boundary conditions or parameter values required to achieve a certain function of the thermal system. This allows the apparatus to be used to establish the appropriate process parameters and boundary conditions for the thermal system modeled. In accordance with the instant invention, the inverse analysis can be directly carried out by the modeling apparatus rather than using the conventional approach, which merely solves the direct problem repeatedly, in a lengthy and costly iterative process, to determine appropriate input parameters to achieve a desired result. In other words, in accordance with the instant invention, once a particular thermal process is modeled for a particular set of control parameters, the device may then be used to automatically obtain the necessary control parameters to achieve a desired result by providing the modeling apparatus with parameters corresponding to the desired result.

To carry out the inverse analysis, the modeling apparatus **101** includes an inverse parameter input section **104** also connected to input device **103**. A user inputs into the modeling apparatus **101** parameters corresponding to desired results, e.g., desired temperature characteristics of the system, which are stored in memory **108**. The processing unit **110**, under control of modeling program **111**, uses the previously generated model of the thermal system and the parameters held in memory **108** and derives or predicts particular control parameters to meet the constraints entered through the inverse parameter input section **104**. This process is more fully described below in connection with the examples provided.

In the above description, a general inverse analysis is carried out by providing specific parameters of a desired end result. Alternatively, dynamic optimization may be used rather than specifying the desired trajectories as specific constraints. In this manner, the desired results are specified as least-squares inequality constraints. For example, if the system were used in an RTP reactor to control the temperature of a silicon wafer during a reaction process, as more fully described below, instead of requiring exact temperature-time histories at certain points on the wafer (the general inverse analysis problem), the wafer would simply be required to follow a specified temperature trajectory within plus or minus 10 degrees and with no more than 5 degrees temperature variation within the wafer.

Using the inverse analysis and/or the dynamic optimization of the instant invention allows the system to be used as

a powerful design tool. For example, the apparatus may be used to evaluate and understand trajectory feasibility and control authority. For example, if the system were used to design a nominal chemical vapor deposition (CVD) reactor which is to include a maximum heater power of 50 kW and is to be used to carry out a process that calls for a particular temperature ramp to the required process temperature, the inverse model produced by the instant invention can be used to provide power-time histories for the heaters (more fully described below). If the required power exceeds the maximum available for the given design, then the desired trajectory is not feasible with that design. The constraints would either have to be relaxed or an alternate design developed that provided sufficient "control authority."

In accordance with another embodiment of the invention, as further illustrated in FIG. **1**, the modeling apparatus **101** can be used to develop real-time control systems for a particular thermal system. Conventional modeling schemes run "open loop," that is, without feed back. Conversely, nearly all processing equipment is run under "closed loop" process control. Thus, such conventional modeling schemes are not adaptable to use in real-time base development of control routines. In accordance with the instant invention, the modeling apparatus **101** operates sufficiently quick enough to be connected in a closed loop fashion to a control system development tool **120**. In most instances, the control system will be implemented on some sort of computing platform and the control program will be implemented in software. However, the instant invention could be used in connection with other types of control systems which are implemented in hardware, for example. The control processor/development tool **120** is comprised of a computer having software development tools loaded thereon. The computer may also be used to control an actual thermal system **130**. Of course, separate computers could be used for these two functions. When the modeling apparatus **101** of the instant invention is connected to the control system development tool **120**, concurrent engineering of equipment design and control programs can be carried out. In this manner, the thermal system may be modeled in the modeling apparatus **101**, and physically based simulations may be run under the control of the very same process-control software that would eventually be loaded into the actual controllers of the thermal system **130** being concurrently designed in real-time (i.e., on the time scale of the actual thermal process).

As illustrated in FIG. **1**, the modeling apparatus **101** is connected to the control system development tool **120** via an interface **116**. In the control system development tool **120**, a control program is loaded and run to control the thermal system generated by the modeling apparatus **101**. The modeling apparatus **101** and interface **116** emulate or appear to the control program as an actual thermal system. When the control processor **120** is connected to an actual thermal system **130**, the system operates in a closed loop. For example, the control processor **120** controls the thermal radiation source element in the system (i.e., the time/intensity characteristics) by providing control signals along line **124**. The thermal system **130** may include temperature sensors (not shown), the output of which is used to return actual measured data to the control processor **120** along line **124**. The control processor **120** uses the measured temperatures to adjust the control parameters. In conventional systems, while the control program in the control processor may then be altered on the basis of the actual refined temperatures, since the thermal system is already constructed, the freedom to modify the thermal system design is significant limited.



In contrast, the instant invention can be used to concurrently develop the control program and the actual thermal system **130** design. Via the interface **116**, the modeling apparatus provides data to the control development tool **120** which is seen as actual closed loop measured data. In other words, certain dependent variables in the model predicted by the modelling apparatus become the "sensors" (e.g., the temperature measurements returned to the control processor **120**) and boundary conditions in the model become the "actuators" (e.g., the power or temperature intensities of heater elements provided as characteristics via line **117**), which are controlled by the control software in the control development tool.

Since the two elements are connected by communication interface **116**, the controller software and the physical model may be simultaneously executed on different computer platforms. In this manner, both the design of the thermal system (via the input thermal system characteristics provided to the modeling apparatus **101**) and the control system (by changing the control program software) may be concurrently designed. Real time control program development can be carried out in the instant invention since the instant invention can rapidly generate an accurate model of the thermal system, including wavelength dependence. In other words, the modeling apparatus of the instant invention generates its model sufficiently quick enough, that it acts as though an actual thermal system.

Below is provided a more specific example of an embodiment of the instant invention used to design and/or control a rapid-thermal-processing (RTP) reactor. RTP reactors are used in silicon-semiconductor fabrication and provide a convenient illustration of the principles of the instant invention and the manner in which the various elements are constructed and used to resolve specific reactor-design and process-optimization problems. Specifically, RTP provides a convenient illustration of the effects of wavelength-dependent (spectral) thermal-radiation transport.

Spectral-radiation behavior is an important consideration in semiconductor processing equipment, especially in lamp-heated systems that have partially transmitting components like quartz windows and the silicon wafer itself. A cross-section of an RTP-reactor **201**, is shown in FIG. **2**. The window **202**, separates the lamp housing **204** from the reaction chamber **206**. The lamp housing **204** houses a number of lamp rings **205** which are individually controlled sources of radiant energy. A silicon wafer **208** is provided in the reaction chamber **206** and is supported by the support members **209** which are fixed to the inner walls **210** of the reaction chamber **206**. In the modeling apparatus **101** (FIG. **1**), the window **202** and the wafer **208** are represented by a radial one-dimensional transient thermal energy equation that describes thermal conduction within the wafer **208** and window **202**, convective heat transfer at top and bottom faces thereof, and thermal radiation exchange between the window **202**, the wafer **208**, and the other features in the system, including the lamp housing and reactor walls **212**, **210**. The second component of the model is the spectral radiation model. In accordance with the instant invention, the complex radiation exchange between all the surfaces in the reactor, including internal reflections **214** within partially transmitting media (e.g. window **202**) are taken into account by the modeling apparatus **101** (FIG. **1**). The instant invention considers the radiative energy transport in wavelength bands. The radiation model produced by the modeling apparatus provides net radiative fluxes to the top and bottom surfaces of the wafer **214** and window **202**, as well as internal heating due to energy absorption. Coupling between

the components is accomplished via the radiative energy fluxes to the window **202** and wafer **208**.

In the instant invention, the differential-equation part of the model parameters is essentially one dimensional. The modeling apparatus **101** executes quickly, even when the processing unit **110** is implemented on workstation class computing platforms. Thus, the modeling apparatus **101** quickly accounts for spectral-radiation effects, and, as described above, may be used in design and real-time control systems.

In order to better understand the instant invention, the underlying principles used by the modeling apparatus **101** to produce a model of a thermal system will now be described. For radiation heat flux at each surface in the thermal system, radiative heat exchange in each wavelength band is treated individually. The contribution of all wavelengths are then integrated into a compound radiation energy. The processed spectral radiation model is fully three-dimensional in so far as the geometric view factors consider three-dimensional geometries. While complex geometries for the radiation exchange are considered, the overall processing is simplified by considering only one-dimensional energy transport in the wafer **208** and window **202**.

While the processing unit **110** (FIG. **1**) treats spectral radiation exchange in great detail, it does not directly consider specular reflections. A diffuse reflector is a surface whose bi-directional reflectivity is independent of reflectance angle; and a diffuse emitter-absorber is a surface whose directional reflectivity (absorptivity or emissivity) is independent of incidence angle. The assumption of radiation diffusely emitting and reflecting surfaces is reasonable since many reflections and re-reflections within an enclosure tend to average out radiative nonuniformities even when there are very smooth surfaces in the enclosure. For many practical problems, the diffuse analysis is actually valid and will give satisfactory results. The assumptions of diffuse surfaces is not valid for the enclosure with very "irregular" geometry such as long, narrow cracks with smooth surfaces, which need to be treated as specular reflectors by special approaches, such as Monte-Carlo or ray-tracing methods.

The processing unit **110**, under control of the modeling program **111**, calculates a model from the input parameters corresponding to the thermal system to be modeled by the modeling apparatus **101**. The following discussion defines the underlying principles and calculations used by the modeling apparatus to calculate the exchange of radiant thermal energy. The principles described below are more fully described in Aili Ting, "*The Influence of Wavelength-Dependent Radiation in Simulation of Lamp-Heated Rapid Thermal Processing Systems*," (2nd International Rapid Thermal Processing Conference, RTP '94, Round Rock, Tex., 1994) pp. 102–109, incorporate herein by reference.

Net heat flux per unit wavelength λ, per unit area at each surface is:

$$\left(\frac{dq(\lambda)}{d\lambda}\right)_{net} = \left(\frac{dq(\lambda)}{d\lambda}\right)_{incoming} - \left(\frac{dq(\lambda)}{d\lambda}\right)_{emission} - \left(\frac{dq(\lambda)}{d\lambda}\right)_{reflection} - \left(\frac{dq(\lambda)}{d\lambda}\right)_{transmission} = \left(\frac{dq(\lambda)}{d\lambda}\right)_{absorbion} - \left(\frac{dq(\lambda)}{d\lambda}\right)_{emission}$$

The incoming flux per unit for all opaque or non-opaque surface elements is described by a vector-matrix equation resulting from a thermal balance inside the reactor, as

$$\left(\frac{dq(\lambda)}{d\lambda}\right)_{incoming} = |I - S\,diag\,R_e(\lambda) - S_1\,diag\,T_r(\lambda)|^{-1} * S\,diag(E(\lambda)\,e_{\lambda b}(\lambda,T)$$

where I is the identity matrix, T is temperature, S is the view factor matrix, $S_1$ is related to S and will be explained later, Re is the reflection matrix, and Tr is the transmission matrix. Planck's spectral distribution of emissive power for a blackbody is given as

$$e_{\lambda b}(\lambda, T) = \frac{2\pi C_1}{\lambda^5\left(\exp\left(\frac{C_2}{\lambda T}\right) - 1\right)},$$

where $C_1$ and $C_2$ are Planck's constants.

The reflection matrix $R_e(\lambda)$ is diagonal matrix that represents the fraction of incoming energy reflected as a result of multiple internal reflections,

$$R_e(\lambda) = \rho(\lambda)\left[1 + \frac{(1-\rho(\lambda))^2(\tau(\lambda))^2}{1-(\rho(\lambda))^2(\tau(\lambda))^2}\right],$$

where ρ is reflectivity and τ is transmissivity. For an opaque surface, $R_e(\lambda)$=diag ρ(λ)=I–diag ε(λ), where ε(λ) is the emissivity. The transmission matrix $T_r(\lambda)$ is a diagonal matrix that represents the fraction of incoming energy that is transmitted as a result of multiple reflections in a non opaque media,

$$T_r(\lambda) = \frac{(1-(\rho(\lambda)))^2\tau(\lambda)}{1-(\rho(\lambda))^2(\tau(\lambda))^2}.$$

$T_r$ becomes a zero matrix for an opaque surface. The absorption or emission matrix E is a diagonal matrix that represents the fraction of incoming energy absorbed as the result of multiple reflections in a non opaque media,

$$E(\lambda) = \frac{(1-\rho(\lambda))(1-\tau(\lambda))}{1-\rho(\lambda)\tau(\lambda)},$$

E(λ)=diagε(λ) for an opaque surface. Because ρ(λ)+τ(λ)+ε(λ)≡1, the above expression satisfies the equation:

$$R_e(\lambda)+T_r(\lambda)+E(\lambda)\equiv I.$$

$S_1$ is a matrix related to the view factor matrix S in such a way that $(S_1)_{ij}=(S)_{i,j1}$. This provides a method to predict radiation exchange between surfaces separated by a semi-transparent material (i.e., quartz window **202**). The subscripts j and $j_1$ represent surface elements on opposite sides of the RTP quartz window.

Integrating over all wavelengths, the total heat flux to each surface is

$$Q_{rad}(T) = \int_0^{\lambda max} diagA\{diagE(\lambda)\,[I - S\,diagR_e(\lambda) - S_1\,diagT_r(\lambda)]^{-1}\,S - I\} * diag[E(\lambda)e_{\lambda b}(\lambda, T)]d\lambda + diagA\{diagE(\lambda_{max})\,[I - S(I - diag\epsilon(\lambda_{max}))]^{-1}S - I\} * diagE(\lambda_{max})(I - F)\sigma T^4$$

where A is the diagonal matrix of surface element area, and

$$F=\frac{1}{\sigma T^4}\int_0^{\lambda_{max}} e_{\lambda b}(\lambda, T)d\lambda$$

is the fractional blackbody emission, and $\lambda_{max}$ is the wavelength after which the surface becomes opaque, $\tau$=O, and $\epsilon$=constant.

Numerically, the integrals are evaluated by using the conventional trapezoidal rule,

$$\int_0^{\lambda_{max}} A\, d\lambda=\frac{1}{2}\sum_{n=1}^{N-1}(A_n+A_{n+1})(\lambda_{n+1}-\lambda_n),$$

where N is the maximum wavelength grid number for $\lambda_{max}$. The equation representing the vector of net radiative fluxes to all surface elements, $Q_{rad}(T)$ can be interpreted physically

$$Q_{rad}(T)=$$

absorbed flux

total incoming flux, including reflection and transmission

emission from all surfaces

$$\int_0^{\infty} diagA\ diagE(\lambda)[I-SdiagR_e(\lambda)-S_1diagT_r(\lambda)]^{-1}SdiagE(\lambda)e_{\lambda b}(\lambda, T)\,d\lambda-\int_0^{\infty} diagA\ \underbrace{diag(E(\lambda)e_{\lambda b}(\lambda, T))d\lambda}_{\text{emitted flux}}$$

The above banded wavelength-dependent radiation model is general. Thus, the modeling apparatus **101** can be used to evaluate and predict wavelength-dependent thermal transport in any dimensional thermal system. Radiation heat flux at surfaces generally provides the surface thermal boundary conditions of the system. Since the instant invention considers only one-dimensional (radial) thermal conduction in the wafer **208**, support **209** and window **202**, the thermal balance forms a one-dimensional partial differential equation (PDE) system with radiation heat flux at the surface serving as a heat source term in the equation for the element on which the surface resides. The thermal balance also involves thermal convection at the surfaces of the wafer **208** and window **202** due to flowing gas which forms another heat source term in the one-dimensional thermal energy equation. Therefore, the matrix-form of the thermal energy equation for all elements of the wafer **208**, support **209**, and window **202** can be treated as

$$C\frac{\partial T}{\partial t}=Q'_{rad}(T)+Q_{cond}(T)+Q_{conv}(T),$$

$$Q'_{rad}(T)=Q_{rad}^{top}(T)+Q_{rad}^{bottom}(T)(+Q_{rad}^{edge}(T)),$$

where C is the element capacity vector, $C=\rho c_p V$, where $\rho$ is the mass density, $c_p$ is specific heat, and V is the volume. The radiation contribution at the top and bottom surfaces is evaluated from the banded radiation model. The radiation heat flux at the wafer edge gives a boundary condition for cases without the wafer support **209**.

The radial conduction is represented by the discrete (finite difference) form of

$$Q_{cond}(T)=\frac{\partial}{\partial T}\left(k(T)A(r)\frac{\partial T}{\partial r}\right),$$

where A(r) is the cross-sectional area of the window **202** or wafer **208** and k(T) is the thermal conductivity. The discrete form consists of a tri-diagonal matrix multiplied by the temperature vector. Symmetric boundary conditions are applied at the center of the wafer **208** and window **202**. Conducting or adiabatic boundary conditions are applied at the support **209** and wall **210** interface if there is a ring-support and radiation heat flux is applied at wafer edge if the wafer is supported by pins. A thermal contact resistance or radiation heat flux is applied at the wafer **208** and support **209** interface for the ring-support.

Convection at the top and bottom surfaces is represented by

$$Q_{conv}(T)=-h(p,T)A_s(r)(T-T_{gas}),$$

where $A_s(r)$ is the surface area, h(p,T) is the pressure- and temperature-dependent convection coefficient. $T_{gas}$ is the temperature of the gas. The discrete form of the convection term consists of a diagonal matrix multiplied by a temperature vector.

As more fully described below, the modeling apparatus can be used to first model the spectral radiation transport in the RTP reactor **201** and then to receive a prescribed wafer time-dependent temperature profile and provide the necessary time-dependent lamp intensity control to achieve the proscribed profile. For this purpose, the instant invention treats the lamps **205** as a surfaces in the lamp housing **204**. A relationship between the lamp temperature and the applied lamp power can be readily derived. The lamp power and temperature are related as follows,

$$P=A\sigma(T^4-T^4_{wall})$$

where P is the lamp **205** power, A is the lamp **205** area, $\sigma$ is the Stefan-Boltzmann constant, T is the lamp **205** temperature and $T_{wall}$ is the chamber wall **210** temperature. For the simple case that the lamps are flat cylindrical rings at the top of the chamber, as illustrated in FIG. **2**, the area of each lamp ring $A=\pi(R^2_o-r^2_i)$, where $r_o$ and $r_i$ are the outer and inner radii of the lamp ring, respectively.

The above described PDE may be solved by modeling apparatus **101** using known techniques such as DASSL software described more fully in R. J. Kee, L. Petzold, *A Differential/Algebraic Equation Formulation of the Method-of-Lines Solution to Systems of Partial Differential Equations*, Sandia National Laboratories Report, SAND82–8893 (1986), L. R. Petzold, *A Description of DASSL: A Differential/Algebraic Solver* Sandia National Laboratories

Report, SAND82-8637 (1982), and K. Brennan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, New York: North Holland (1989), the contents of which are incorporated herein by reference. In general, the modeling apparatus uses a general method-of-lines approach to convert the PDE to an ordinary differential equation (ODE). DASSL is designed to solve stiff differential-algebraic systems (DAE) of the form $g(t, y, y')=0$, rather than $y'=f(t, y)$. DASSL is a variable-order method based on the backward-differentiation-formula (BDF) methods. DASSL uses error-estimation methods to choose a sequence of time steps that controls local truncation error.

The modeling apparatus of the instant invention was tested for use in design and control of an RTP reactor as described below. In the examples provided, 11 radial elements are used each to represent the wafer **208** and the window **202**.

The reactor **201** radius is 12.0 cm, the lamp-to-window separation is 4.9 cm, the window-to-wafer separation is 0.165 cm, and the wafer-to-showerhead (located at the bottom of the reactor and not shown) is 3.43 cm. A quartz window **202** of thickness 12.7 cm is provided and a silicon wafer **208** of thickness 0.635 mm is provided. There are four lamp **205** rings at the top of the reactor having center radii of 2.223 cm, 5.08 cm, 7.84 cm, and 10.9 cm. Each lamp **205** ring has a width of 1.5 cm. A graphite wafer support **209** spans the distance from the wafer edge to the reactor wall **210**. The view factors for such a system can be calculated analytically. The modeling apparatus of the instant invention is capable of predicting radiation effects in three dimensional geometries for such an RTP reactor.

The mass densities of the wafer **208**, window **202**, and support **209** are 2330, 2200, and 7801 kg/m$^3$ respectively. The thermal conductivity of the window is held constant at 744.8 J/kg-K. FIGS. **3A–3H** illustrate graphically the proportions of materials used in the test of the modeling apparatus. FIGS. **3A** and **3B** show the specific heats and temperature-dependent conductiveness for silicon and graphite. The band-dependent emissivity of stainless-steel is shown in FIG. **3C**. The emissivity of the graphite support is fixed at 0.95. We use a fixed emissivity of 0.05 to represent the reflector material at the top of the chamber. We take the emissivity of the lamps to be that of tungsten, $\epsilon$=0.45 or as a function of wavelength as shown in FIG. **3D**. FIG. **3E** shows the transmissivity and refractive index for quartz. This data can be used to determine the wavelength-dependent quartz emissivity, (in one test the quartz emissivity was determined to be $0.8 * (1-\tau)$, in another it is fixed at $(1-\tau-\rho)$ where reflectivity is taken as 0.05). The wavelength-dependent transmissivity of quartz is also shown in FIG. **3E**. The wavelength- and temperature-dependent silicon-wafer emissivity is shown in FIG. **3F**. FIG. **3G** shows the wavelength- and temperature-dependent silicon-wafer transmissivity. FIG. **3H** illustrates an approximation where the emissivity of the wafer is only a function of temperature.

Each of the tests (simulations) carried out using the modeling apparatus **101** to predict the behavior of an RTP reactor used 14 bands that are defined to capture features of the wavelength-dependent material properties. Specifically, the bands were divided at wavelengths of 0, 0.17, 0.3, 0.4, 0.6, 1.1, 2.3, 2.6, 3.2, 4.0, 4.2, 8.0, 15.0, and $\infty\mu$m. In spite of the fact that the window is effectively opaque beyond 4.2 $\mu$m, contributions to the wafer response from the longer wavelengths can be observed.

The output results of the modeling apparatus of the instant invention were compared to data from tests carried out on actual RTP reactors, where step power changes were applied to individual lamp rings. FIG. **4** shows data from one such test compared to predictions output by the modeling apparatus **101** of the instant invention. In this test only zone two, the maximum power of which is 11 kW, was varied. From 0 to 67 seconds 20% power was applied, from 67 to 127 seconds the power was 30%, from 127 to 190 seconds the power was 40%, from 190 to 250 seconds the power was 50%, from 250 to 324 the power was 60%, and from 324 to 364 seconds the power was 70%. The initial temperature for the entire system was 300° C. The modeling apparatus maintains wall temperatures at 27° C. throughout, although there is some ambiguity about the experimental wall temperatures at the start of the tests. Further, the modeling apparatus applied no convective heat transfer on either the wafer **208** or the window **202**. An interface resistance of 0.1 W/m-K between the wafer **208** and the support **209** was used.

As illustrated in FIG. **4**, a comparison of the modeling apparatus **101** predictions to the wafer center (upper curve) and edge temperature as measured by wafer-embedded thermocouples shows strong correlation of actual behavior. Initially, there is an offset between the predicted and measured temperatures, but the comparison improves near the operating temperatures of the RTP reactor. Some of the disagreement at early times is due to uncertainty about the starting conditions in the experiment. The test demonstrates a degree of agreement between the model predictions and the actual data which is quite satisfactory. Thus, the results of the modeling apparatus can be used with confidence to predict effects of various approximations in the radiation transport and to facilitate the design of actual thermal systems.

FIGS. **5** and **6** illustrate radial temperature profiles in the wafer **209** at an instant in time. These profiles were generated using the modeling apparatus **101** to demonstrate the sensitivity of the modeling apparatus **101** to physical approximations. In other words, the modeling apparatus was used to predict the spectral radiation transport for a number of different thermal system characteristics. In each of the illustrated profiles, the input power settings were the same.

FIG. **5A** illustrates the effect of allowing the silicon wafer **208** to be semitransparent, permitting transmission of the medium wavelength energy at low temperature. FIG. **5B** compares input characteristics using a full wavelength-dependent window and a window with constant emissivity and wavelength dependant transmissivity. FIG. **5C** illustrates the effect of banded wavelength radiation at the stainless-steel reactor-chamber walls **210**. FIG. **5D** contrasts the difference between the cases with and without convection at the wafer **208** surface.

FIG. **6A** compares single band and banded wavelength-dependent radiation. FIG. **6B** compares conducting and insulated boundary conditions at the wafer support **209** and wall **210** interface. FIG. **6C** illustrates normalized radiation heat fluxes over the wafer **208**. FIG. **6D** illustrates the normalized, banded wavelength-dependent, incoming heat flux to the top surface of the wafer **208** center.

FIGS. **5** and **6** illustrate the effect that different approximations have on the predicted wafer temperature. The magnitude of temperature differences depend on the radiative properties applied for the different cases. The fluxes illustrated in FIGS. **6C** and **6D** provide qualitative and quantitative views of how radiative heat is distributed on a wafer and over wavelength. Further, they illustrate how the effect of varying the input characteristic which corresponds to varying the proposed design of the RTP reactor can be

immediately observed in the output of the modeling apparatus.

Using the above example of an RTP reactor, an example of how the inverse analysis can be implemented in the modeling apparatus of the instant invention is described. The transient trajectory of the wafer temperature is input into the modeling apparatus **101** as inverse parameters. Using these parameters, and the previously generated model, the lamp power trajectories required to meet the desired wafer trajectory can be calculated by converting the parabolic initial-boundary value problem into a differential-algebraic problem.

In the four-lamp-zone reactor illustrated in FIG. **2**, the desired temperature trajectories at four points on the wafer **208** are specified. In addition to the residual equation resulting from the energy balance at each point on the wafer **208**, as obtained from the initial prediction of the modeling apparatus, the specified trajectory introduces a new residual equation, $F_i=T_i-T_{set}(t)$. Since there are now two residual equations at four points on the wafer (the energy-equation residual and the trajectory constraint), the problem would be mathematically over specified unless new dependent variables were introduced. The modeling apparatus **101** selects the lamp powers as the needed dependent variables. Thus, rather than specifying the lamp powers as parameters or boundary conditions (the direct problem), the modeling apparatus **101** determines, as a function of time, the required lamp power input (the inverse problem) and outputs the results.

The direct problem can be solved without difficulty by a variety of methods, including the method-of-lines. The inverse problem, however, can cause great difficulty for traditional computational methods. One problem is that most ordinary-differential-equation software requires problem specification in the "standard form," where each residual equation must contain the time derivative of a dependent variable. The trajectory constraint equations do not fit that form. The algorithms that underpin the DASSL software can handle the problem specification and the solution, however, for the above described problems.

FIG. **7** illustrates the output results of an inverse operation of the modeling apparatus **101** of the instant invention. The specified wafer trajectory **702** (shown as a heavy line), and the derived required lamp powers **704**, **705**, **706**, **707** are shown as a function of time. As expected the lamp powers increase initially to drive the wafer temperature up according to the trajectory. After peaking at about 5 seconds, however, the lamp powers decrease even though the wafer temperature continues to increase. This behavior is caused by the fact that the silicon-wafer-infra-red-transmissivity changes rapidly around 600° C. from relatively high transmission to essentially opaque. Thus, because the radiation is coupled more effectively to heat the wafer, the lamps must reduce power to maintain the desired wafer trajectory.

As illustrated in FIG. **7**, the inverse output can be used to provide control parameters in terms ofttimes and intensity for powering the heat lamps. In other words, for a desired wafer-temperature ramp, the inverse analysis predicts the required lamp power or cooling capacity. If the required lamp power exceeds that available in a given design, then the trajectory is infeasible, regardless of the controller strategy. Thus, the reactor must be redesigned or a less-aggressive trajectory considered. Inverse analysis also provides information on "control authority," which relates to the ability of the actuators to respond quickly enough. Lamps typically have very fast response time, but resistive heaters or cooling systems may not have the dynamic response needed to accomplish certain trajectories. This information can be used to specify actual reactor designs and as feedback in developing control programs.

The instant invention has been described above in connection with specific embodiments. The principles of the invention are not so limited. Many variations on the uses of the instant invention will be apparent from the preceding disclosure. Accordingly, the invention is only limited by the appended claims.

What is claimed is:

**1**. A method of controlling a rapid thermal processing (RTP) reactor for processing a silicon wafer under a desired wafer time dependent temperature profile, the RTP reactor including at least one heating element and a partially transmitting component separating the heating element and the wafer, the method comprising the steps of:

generating in a computer a spectral thermal radiation transport model of the RTP reactor for given heating element parameters;

inputting into the computer data specifying the desired wafer time dependent temperature profile;

selecting components for the RTP reactor based on the generated model;

calculating an inverse of the generated model using the data specifying the desired wafer time dependent temperature profile to determine heating element parameters required to produce the desired wafer time dependent temperature profile,

wherein the selecting step selects the components for the RTP reactor to meet control parameters indicated by the inverse of the generated model; and

controlling the heating elements of the RTP reactor in accordance with the heating element parameters to heat the wafer in accordance with the desired wafer time dependent temperature profile.

**2**. A method as recited in claim **1**, wherein the inputting step comprises the steps of:

selecting a number of prescribed locations in the wafer; and

assigning each of the prescribed locations a corresponding temperature time dependence as the data specifying the desired wafer time dependent temperature profile.

**3**. A method of designing a rapid thermal processing (RTP) reactor for processing a silicon wafer, the RTP reactor including at least one lamp heating element and a partially transmitting component separating the lamp heating element and the wafer, the method comprising the steps of:

inputting data into a computer corresponding to a general design of the RTP reactor including at least a thermal transport characteristic of the partially transmitting component and the wafer and a general design of the thermal radiation characteristic of the lamp heating element;

generating in the computer a model of spectral thermal radiation transport of the RTP reactor based on the input data;

selecting components for the RTP reactor based on the model;

inputting data corresponding to a desired time dependent temperature profile of the wafer to be processed in the RTP reactor;

storing the data corresponding to the desired time dependent temperature profile in a memory;

retrieving the stored data and utilizing the computer to calculate an inverse of the model to obtain control

parameters for the heating element necessary to heat the wafer in accordance with the retrieved data; and

providing a control portion to control the heating element based on the control parameters obtained from the inverse of the model.

**4**. A method as recited in claim **3**, further comprising the step of utilizing the computer to calculate an inverse of the model, wherein the selecting step includes the step of selecting components for the RTP reactor to meet control parameters indicated by the inverse of the model.

**5**. A method of designing a rapid thermal processing (RTP) reactor for processing a silicon wafer, the RTP reactor to include at least one lamp heating element and a partially transmitting component separation the lamp heating element and the wafer, the method comprising the steps of:

inputting data into a computer corresponding to a general design of the RTP reactor including at least the thermal transport characteristic of the partially transmitting component and the wafer and the general design of the thermal radiation characteristic of the lamp heating element;

generating in the computer a model of spectral thermal radiation transport of the RTP reactor based on the input data; and

selecting components for the RTP reactor based on the model;

utilizing the computer to calculate an inverse of the model to obtain control parameters for the heating element necessary to heat the wafer in accordance with a desired time dependent temperature profile of the wafer to be processed in the RTP reactor; and

providing a control portion to control the heating element based on the control parameters obtained from the inverse of the model,

wherein the selecting step includes the step of selecting the components for the RTP reactor to meet the control parameters indicated by the inverse of the model.

**6**. A method as recited in claim **5**, wherein the utilizing step comprises the steps of:

selecting a number of prescribed locations in the wafer;

assigning each of the prescribed locations a corresponding temperature time dependance; and

determining the control parameters for the heating element as a function of the corresponding temperature time dependance constrained by the model of the spectral thermal radiation transport of the RTP reactor.

**7**. A system including a modeling apparatus for accurately characterizing time dependent spectral thermal radiation transport of a thermal system, the thermal system including a first portion having one or more heating elements, and a second portion separated from the first portion by a partially transmitting medium, the modeling apparatus comprising:

an input device for inputting characteristics of the thermal system;

a memory connected to the input device for storing the thermal system characteristics, the thermal system characteristics including at least geometric parameters of the thermal system, a time dependent intensity profile of the heating elements and wavelength-dependent properties of the partially transmitting medium;

a processing unit connected to the memory, the processing unit predicting a time dependant spectral thermal radiation transport characteristic of the thermal system based on the thermal system characteristics,

said processing unit producing a characteristic model of the thermal system using the time dependant spectral thermal radiation transport characteristic.

**8**. A system as recited in claim **7**, wherein the modeling apparatus further comprises:

a second input device for inputting inverse parameters specifying a desired thermal transport characteristic of the thermal system

wherein said processing unit is coupled to said second input and is configured to produce an inverse of the characteristic model, the inverse of the characteristic model indicating a characteristic of the thermal system which is necessary to produce the desired thermal transport characteristic.

**9**. A system as recited in claim **8**, wherein the desired thermal transport characteristic is a time dependent temperature at a location within the second portion of the thermal system and the characteristic of the thermal system which is necessary to produce the desired thermal transport characteristic is a time dependent intensity profile of the heating elements.

**10**. The system as recited in claim **7**, further comprising:

developing means coupled to the modeling apparatus for developing a control program for controlling the heating elements of the thermal system, the developing means receiving from the modeling apparatus temperature indications corresponding to a modeled temperature at selected locations within the characteristic model of the thermal system and providing to the modeling apparatus the time dependent intensity profile of the heating elements on a basis of heating element control parameters indicated by the control program; add

input means coupled to the developing means for modifying the control program on a basis of an output from the modeling apparatus.

**11**. The system as recited in claim **10**, further comprising interface means connected between the developing means and the modeling apparatus for controlling exchange of information between the developing means and the modeling apparatus.

**12**. The system as recited in claim **11**, wherein the developing means and the modeling apparatus are simultaneously run on separate computing platforms and the interface means emulates an actual thermal system in real-time using an output from the modeling apparatus for the development of the control program.

**13**. The system as recited in claim **10**, wherein the thermal system is coupled to the developing means, wherein the developing means receives actual real-time data from the thermal system which is used by the developing means to adjust the time dependent heating profile of the heating elements that is provided to the modeling apparatus, and wherein the system is a closed loop system.

* * * * *

3. Su-shing Chen, "AEMPES: An expert system for in-situ diagnostics and process monitoring," See Office Action of February 7, 2008.

**AEMPES: An expert system for in-situ diagnostics and process monitoring**

Su-shing Chen

University of North Carolina-Charlotte
Charlotte, North Carolina 28223

**Abstract**

An expert system - ***AEMPES*** (Advanced Electronic Materials Processing Expert System) for in-situ diagnostics and process monitoring, is being developed. This expert system is a key component of the ***intelligent manufacturing equipment architecture*** which is proposed to integrate the manufacturing line with its simulator. In the expert system, there are two interrelated subsystems - a ***neural network*** subsystem for adaptive process control, monitoring, and learning, and a ***rule-based*** subsystem for human interface and high-level AI reasoning.

## 1. Introduction

An intelligent manufacturing equipment architecture was defined in [2]. In [5], [10], an idea to implement this architecture was proposed. It has two components:

(1) manufacturing equipment - equipment hardware and sensors.
(2) intelligent expert workstation - a hybrid AI/neural network expert workstation.

For the intelligent expert workstation, we propose the philosophy of a hybrid AI/neural network expert workstation. Its model-based reasoning scheme contains process models and the equipment model. A neural network simulation software is used to learn process models and the equipment model (Sometimes, human expert knowledge is very useful to prescribe the network topology so that the learning is supervised). A neural network hardware emulates the process models and the equipment model for adaptive and on-line process monitoring and control. A rule-based expert system provides human interface and high-level decision support.

We have discovered that a manufacturing line can be integrated with its manufacturing line simulator at the intelligent expert workstation level. An intelligent expert workstation provides a link between the manufacturing line and its simulator at the equipment module. In essence, the manufacturing line is distributed into clusters of processing modules, each of which has an intelligent expert workstation described above. At present, manufacturing (hardware) lines and their (software) simulators are separately functioning [1].

The organization of this paper is as follows. In section 2, some important issues of manufacturing automation are discussed. In section 3, the distributed AEMPES system is described. We refer to [7] for a related result on group technology in semiconductor design for manufacturing. In section 4, the intelligent equipment architecture is proposed. In section 5, the ideas of implementing an individual AEMPES expert workstation are presented. In addition to the neural network methodology, qualitative simulation, discussed in [6], is also quite useful in the model-based reasoning. In section 6, a neural network software - INNSE (Interactive Neural Network Simulation Environment) is presented. INNSE is a subset of the expert workstation. More detailed modeling results of processes and equipment will appear elsewhere.

## 2. Manufacturing Automation

In manufacturing automation, two major objectives are the tight coupling of CAD/CAM/CAT and the integration of a manufacturing line and its simulator [1], [2]. First, in manufacturing automation, the design, manufacturing and testing should not be separate stages. There are at least two reasons:

(1) The flow of the complete fabrication process relies on the synchronization of many steps in the three stages.
(2) The tight coupling of three stages enables the control, scheduling, and management of the complete fabrication process to be more effective and adaptive.

The tight coupling requires simultaneous considerations of manufacturing process design and relevant device design, an enormous endeavor. At present, we do not know exactly how to couple the three stages tightly. However, the starting point should be the modeling of equipment and processes and the interface of equipment/process models with CAD and CAT data.

Secondly, the integration of a manufacturing line and its simulator is essential to autonomous manufacturing systems, because the manufacturing line simulator may be considered as the "brain" which maintains the central intelligence of a manufacturing system. The autonomous (human like) behavior of a manufacturing system is manifested by the level of integration of its "brain" and its "arms and legs".

The integration enables adaptive sensor-driven control, learning, planning, and optimization in production. At present, we do not have complete integration of manufacturing lines with their simulators. The main reason is the lack of a linking point between the two.

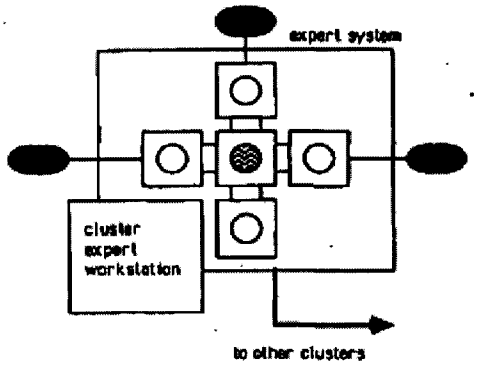
An equipment model is the computational model of a semiconductor fabrication equipment, such as an optical lithography stepper or an ion implanter. Because of the diversity of semiconductor fabrication equipment, their models should have an open architecture that is developable using rapid prototyping techniques. Moreover, these models should be multifunctional for assisting manufacturing engineers, process engineers and operators.

## 3. Distributed System - AEMPES

The logical architecture of AEMPES depends on the clustering of process modules and the interconnection of different clusters. A vacuum wafer transport system is being developed for interconnecting different clusters. Within each cluster, processing modules are completely integrated in the Brooks mechanism. Providing each processing module with an expert system, we require a central expert workstation composing of these individual expert systems. The characterization, testing and diagnostics measurement and process control are performed in the central expert workstation. Indeed, this is a tightly coupled cluster.

Nontheless, monitoring, control and diagnostics of the collection of all cluster subsystems are quite different. Physically, the local area network interconnecting those subsystems is in parallel with the vacuum wafer transport system. Due to the complexity of semiconductor processing, a centralized explicitly constrained and minimally adaptive approach may not be adequate. A distributed system consisting of interacting and coordinating cluster expert workstation is necessary. Thus, techniques in distributed AI systems are used in the global AEMPES.

CH2810-6/90/0000-0119 $1.00 ©1990 IEEE




Distributed AI mechanisms are quite complex. Some of the issues include:

(1) Accomodation of open systems (systems with no complete representation and with dynamically changing boundaries).
(2) Combinatorial implosion, the possibility that partial results at one cluster in the system may greatly constrain the potential solution space at another cluster,
(3) Adaptability, the possibility that concurrent systems are inherently more adaptable than sequential systems.
(4) Multiple perspectives, the need to make different viewpoints consistent.
(5) Modeling and analysis of coordinating intelligent agents are different from usual methodologies.

Often global system information is needed by clusters to make local decisions because of interdependency of different clusters. There is no universal global optimization criterion. We propose the cooperative Pareto optimization scheme. Intuitively, it is a scheme which optimizes multiple objective functions of different clusters. A solution to the Pareto optimization problem is one which does not allow preferential treatment of any individual cluster. If a certain priority is used in the overall factory planning, this scheme must be modified. Otherwise, the sequencing of processing tasks follows this scheme.

Associated with this distributed AI approach, there are the issues of data communication network standard and databases. The SEMI Equipment Communications Standard (SECS) is extendible to this case. A distributed database fits naturally into the distributed AI setting. In fact, it is a subset of the distributed expert system.

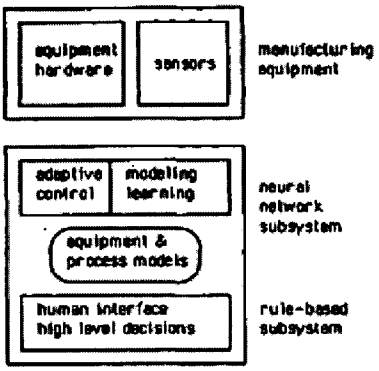
## 4. Intelligent Equipment Architecture

The intelligent equipment architecture consists of two components:

(1) manufacturing equipment - equipment hardware and sensors.
(2) intelligent expert workstation - a hybrid AI/neural network expert workstation.

For the intelligent expert workstation, we propose the philosophy of a hybrid AI/neural network expert workstation. Its model-based reasoning scheme contains process models and the equipment model. A neural network simulation software is used to learn process models and the equipment model. A neural network hardware emulates the process models and the equipment model for adaptive and on-line process monitoring and control. A rule-based expert system provides human interface and high-level decision support.

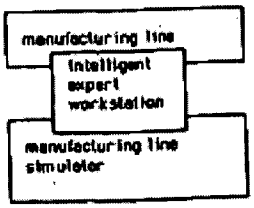
The neural network subsystem which emulates the process/equipment models serves as the "intelligent controller" of the equipment module. The intelligent controller has real-time links with VLSI integrated sensors and actuators of the fabrication equipment. Thus, the manufacturing line and its simulator are tightly integrated in a close-loop manner.



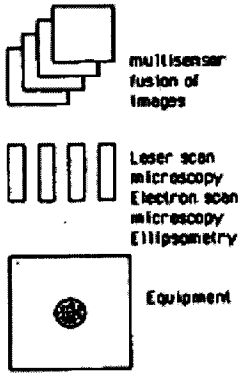

Intelligent equipment architecture

Close-loop feedback control at each equipment module requires in-process inspection, testing and characterization. Laser scan microscopy, electron scan microscopy, ellipsometry, and interferometry could provide in-process data for extraction of spatial information, such as linewidths and thickness, and to inspect particulate-induced defects. The expert workstation has to determine whether a given equipment parameter is within certain specifications from the extracted feature values. We could investigate multisensor fusion so that more complete spatial information is obtained. Furthermore, electrical C-V measurements are fused with sensed image data, and their spatial interpretations.

The learning and optimization capabilities are supported by the neural network subsystem of the intelligent expert workstation. However, a neural network system alone cannot provide complete solutions to complex engineering problems. A hybrid AI/neural network expert system is proposed here to capture manufacturing knowledge, to perform adaptive control, and to make planning and decision support, interacting with the manufacturing simulator.




Integration of manufacturing line and its simulator




5. AEMPES: Real-time Diagnostics and Control

In each cluster, AEMPES is to capture the capability of experienced process engineers for diagnostics, monitoring, and control of the cluster of equipment modules. This experience is integrated with computer analysis capability and basic models of the underlying physics of the cluster of equipment modules. The knowledge required is coded into AEMPES for assisting process operators or autonomously monitor/control the fabrication processes.

In AEMPES, we use the concept of model-based reasoning. Model-based reasoning is an extension of rule-based expert system technology [3]. It is a subfield of knowledge engineering that involves building and analyzing an explicit computational model of the structure, principles and behavior of a system. In a traditional rule-based expert system, heuristics used by an expert to solve a particular problem are coded into the knowledge base. In a model-based system, knowledge is represented explicitly in computational models. Model-based reasoning is more flexible, because the same system model may be used for different applications of a cluster expert workstation:

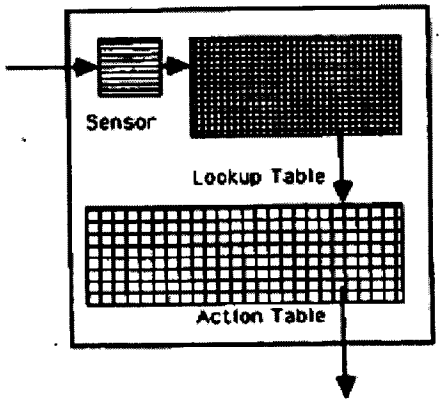
Diagnostics,
Sensor/actuator interface,
CAD/CAT interface,
Human operator interface,
Planning,
Decision support,
Control,
Analysis.

It is possible to integrate different perspectives of various applications in a single model. This increases the power of an expert system. Furthermore, model-based reasoning systems may be extended and modified quite easily to reflect any change in the processing and overall manufacturing environment. Traditional rule-based reasoning systems can only be changed by rewriting a significant amount of codes. Model-based systems are algorithm-based. Algorithms can be revised continuously in computational models by human expert or machine learning (using both AI and neural network approaches).

Conventional model-based reasoning systems are object-oriented [3]. Structural elements of a model may include:

Concepts (modules and submodules)
Objects (sensors, components in a submodule)
Attributes (parameters -temperature, voltage, pressure)
Relations (A is a component of B, B is a subsystem of C)
Functions (thermal oxidation, cleaning)

For example, in the application of diagnostics, functions are represented from the perspective of component faults on the functions. The values of a component attribute determine whether a component is normal or not. Simple programs can be embedded in each breakable component to create a "test-case generator" that automates fault analysis and simulation.




The object oriented trees of model-based reasoning can be represented by neural networks. Conversely, knowledge in neural networks can be represented by semantic networks in object-oriented forms. This has been established in the literature. See also [8,9] and [11,12]. However, in some neural network modeling techniques (e.g. the back propagation technique), additional internal variables are introduced and models involving additional variables are learned. Although these variables may not have immediate physical interpretations, we consider them as essential process/equipment parameters. See the next section for some modeling examples.

6. Interactive Neural Network Simulation Environment

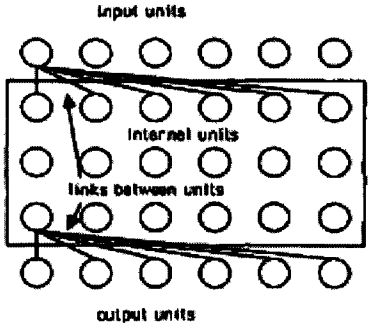
Interactive Neural Network Simulation Environment (INNSE) is a hierarchical neural network simulator, with its macrocells constructed by users, is capable of modeling equipment, processes, and manufacturing lines, and is capable of simulation run as flow of activation networks. As CAD tools for circuit design engineers, INNSE is a part of the manufacturing process design software environment in the intelligent expert workstation described above.

Each neural network model consists of a set of input units and output units, a set of internal units, a transition rule, and a learning rule. There may be as many layers of internal units as we wish.

Let us denote by $(u_i)$ the input units, $(v_j)$ the output units, and $(w_k)$ the internal units. The values to be assumed are between 0 and 1. This requires a scaling factor for each physical variable. The link weights between these units are denoted by $\lambda_{ik}$ and $\lambda_{kj}$, indicating connections between input units, internal units, and output units. The link weights assume values between -1 and +1. A positive link weight indicates exitatory connection, and a negative link weight indicates inhibitory connection.

For illustration, we discuss two examples. The first is a neural network model of the thermal oxidation process. In [13], a power law was proposed for the oxidation thickness:

$$x = at^b,$$




where a and b are functions of the temperature T and the pressure p, and t is the time. A more detailed power law involves with the initial thickness $x_0$ and the initial time $t_0$. It was shown that the power law fits all observed data in thermal oxidation [13]. Easily, we have modeled the functions a and b as output units of a neural network with input units T, p, $x_0$, $t_0$. Using a back propagation learning rule, we can construct this network with 10 internal units within error rate of 1%. This network can predict the behaviors of a and b under input values which have not been experimented. The thickness x can also be determined for unexperimented input values, by either the power law or a neural network of itself (x is treated as a function of T, p, $x_0$, $t_0$). In future work, we shall verify our results with experimental data. If successful, we shall use this network for multiple-stage oxidation analysis and control.

The second example is the neural network modeling of the Balzers SWS 605 system. It is a single wafer, cassette-to-cassette, planar magnetron sputtering system. This model is a first phase modeling result. In the later phases, we plan to incorporate expert knowledge into the modeling process, resulting in more sophisticated neural network models. In the present phase, we consider 2 input units - power and gas pressure, and 5 output units - Aluminum deposition rate, TiW deposition rate, metal film stress, sputter etch uniformity, and sputter etch rate. Of 12 internal units, we have constructed the neural network within error rate of 1 %. This rather simple model provides a basic function of this equipment.

## 7. Conclusion

In this paper, we have presented a framework of the intelligent equipment architecture and a neural network modeling methodology. Neural network representations can be translated into the conventional AI reasoning mechanisms for developing the intelligent expert workstation - AEMPES. The present statistical modeling techniques, such as regression analysis and factorial design, do not have this capability. The more important issue is the real-time, close-loop control in the intelligent equipment architecture. We have proposed the same neural networks to serve as the controller [9].

We would like to express our gratitude to SRC and NSF-ERC for their partial financial support. We are also indebted to Jon Fitch of NCSU for his papers and unpublished data on the Balzers system.

## References

1. R. K. Cavin III, DARPA-SRC Workshop: CIM for Integrated Circuits, MIT, June 1987.
2. D. H. Phillips and R. K. Cavin III, Intelligent semiconductor fabrication equipment, DARPA-SRC Workshop: CIM for Integrated Circuits, MIT, June 3, 1987.
3. S. Hedberg and J. Dynis, Extending expert systems technology: Model-based reasoning, Intellinews, Intellicorp, Vol. 2, No. 2, August 1986.
4. S. Chen, AEMPES: An expert system for in-situ diagnostics and process monitoring, SPIE's 1989 Symposium on Microelectronic Integrated Processing: Growth, monitoring, and control, Santa Clara CA, October 8-13, 1989.
5. S. Chen, Multichamber and in-situ processing system design and control, SPIE's 1989 Symposium on Microelectronic Integrated Processing: Growth, monitoring, and control, Santa Clara CA, October 8-13, 1989.
6. S. Chen, QUASI: An qualitative analysis program of surface and interface for microelectronic materials processing, IEEE Computer Society - SPIE Applications of AI VII Conference, Orlando FL, 1990.
7. S. Chen, Group technology in multichamber and in-situ processing, IEEE Computer Society - SPIE Applications of AI VII Conference, Orlando FL, 1990.
8. S. Chen, CONE: Computational network environment, IBM/UNCC Joint Study Report, 1989.
9. S. Chen and M. Zhang, Adaptive (neural network) control in computer-integrated-manufacturing, IEEE Computer Society - SPIE Applications of Artificial Intelligence VI Conference, Orlando FL, 1988.
10. S. Chen, Application of highly-parallel processing techniques to the development of CIM equipment models, Technical Report for SRC, 1988.
11. C. A. Cruz, W. A. Hanson, and J. Y. Tam, Knowledge processing through flow of activation, IEEE First Annual Int. Conference on Neural Networks, 1987, Vol. II, 343-350.
12. C. A. Cruz, W. A. Hanson, and J. Y. Tam, Neural network emulation hardware design consideratiens, IEEE First Annual Int. Conference on Neural Networks, 1987, Vol. III, 427-434.
13. A. Reisman, E. H. Nicollian, C. K. Williams, and C. J. Merz, The modeling of silicon oxidation from $1x10^{-5}$ to 20 atmospheres, Journal of Electronic Materials, Vol. 16, No. 1, 1987, 45-55.
14. V. Hegemann and J. Fitch, Statistical strategies for optimizing processes: Part I and Part II, TI Technical Journal, January-February 1987, 92-97; March-April 1987, 69-76.

## RELATED PROCEEDINGS APPENDIX

None